Un nouveau modèle de correspondance pour un service de messagerie électronique avancée

Laurent Cailleux

THÈSE / Télécom Bretagne
sous le sceau de l'Université européenne de Bretagne

pour obtenir le grade de Docteur de Télécom Bretagne
En accréditation conjointe avec l'Ecole doctorale Matisse
Mention : informatique

présentée par
Laurent Cailleux

préparée dans le département Réseaux, sécurité et multimédia
Laboratoire Irisa

Un nouveau modèle de correspondance pour un service de messagerie électronique avancée

Thèse soutenue le 12 janvier 2015
Devant le jury composé de :

Francine Krief
Professeure, Enseirb-Matmeca - Talence / présidente

François Charoy
Professeur, TELECOM Nancy / rapporteur

Maryline Laurent
Professeure, TELECOM SudParis / rapporteur

Charles Préaux
Professeur, Ecole Nationale Supérieure d'Ingénieurs de Bretagne Sud – Vannes / examinateur

Benoit Martin
Ingénieur d'études, DGA MI - Bruz / examinateur

Ahmed Bouabdallah
Maître de conférences, Télécom Bretagne / examinateur

Jean-Marie Bonnin
Professeur, Télécom Bretagne / directeur de thèse

Philippe Letellier
Directeur de l'innovation, Institut Mines-Télécom / invité

Damien Roque
Maître de conférences, Institut Supérieur de l'Aéronautique et de l'Espace – Toulouse / invité

n° d'ordre : 2015telb0335

Sous le sceau de l'Université européenne de Bretagne

Télécom Bretagne

En accréditation conjointe avec l'Ecole Doctorale Matisse

Un nouveau modèle de correspondance pour un service de messagerie électronique avancée

---

Thèse de Doctorat

Mention : Informatique

Présentée par **Laurent Cailleux**

Département : RSM

Directeur de thèse : Jean-Marie Bonnin

Soutenue le 12 janvier 2015

Jury :

Mme Francine Krief	Professeure	ENSEIRB-MATMECA	(Présidente)
M. François Charoy	Professeur	TELECOM Nancy	(Rapporteur)
Mme Maryline Laurent	Professeure	TELECOM SudParis	(Rapporteur)
M. Charles Préaux	Professeur	ENSIBS	(Examinateur)
M. Benoit Martin	Ingénieur d'études	DGA MI	(Examinateur)
M. Jean-Marie Bonnin	Professeur	TELECOM Bretagne	(Directeur de thèse)
M. Ahmed Bouabdallah	Maître de Conférences	TELECOM Bretagne	(Encadrant)
M. Philippe Letellier	Directeur de l'Innovation	Institut Mines-Telecom	(Invité)
M. Damien Roque	Maître de Conférences	ISAE	(Invité)

TABLE DES MATIERES

LISTE DES TABLEAUX

LISTE DES ILLUSTRATIONS

LISTE DES ACRONYMES

ABNF Augmented Backus-Naur Form

API Application Programming Interface

ASN.1 Abstract Syntax Notation 1

ADMD Administrative Management Domain

BMTA Boundary Mail Transfer Agent

CEA Correspondence Enforcement Agent

DKIM DomainKeys Identified Mail

DNS Domain Name System

IANA Internet Assigned Numbers Authority

IETF Internet Engineering Task Force

IMAP Internet Message Access Protocol

IMF Internet Message Format

MDA Mail Delivery Agent

MIME Multipurpose Internet Mail Extensions

MMHS Military Message Handling System

MS Message Store

MSA Mail Submission Agent

MTA Mail Transfer Agent

MUA Mail User Agent

NIST National Institute of Standards and Technology

OTAN Organisation du Traité de l'Atlantique Nord

POP Post Office Protocol

RGS Référentiel Générale de Sécurité

SASL Simple Authentication Secure Layer

SMTP Simple Mail Transfer Protocol

STANAG Standard Agreement

TLS Transport Layer Security

UIC User Intention of Communication

URL Uniform Resource Locator

RESUME

Le service de courrier électronique en raison de sa simplicité d'utilisation combinée à son efficacité, a constitué l'un des principaux vecteurs de popularisation d'Internet. Il est devenu un service incontournable dont la richesse s'exprime au travers des usages variés et multiples qu'il autorise (privé, professionnel, administratif, officiel, militaire...). Cependant, toutes les réalisations existantes se réduisent techniquement à la mise en œuvre de politiques globales, compilant de façon statique un ensemble limité de fonctionnalités. Ces approches ne permettent pas au système de s'adapter de façon différenciée aux usages. De plus, le caractère rigide et monolithique de ces politiques peut parfois conduire à l'exécution inutile de traitements coûteux ou à l'impossibilité de satisfaire simultanément des exigences contradictoires.

Nous abordons cette problématique de l'évolution de la messagerie électronique dans le cadre général de la communication interpersonnelle d'un locuteur vers un interlocuteur. Nous identifions l'intention de communication du locuteur, comme un paramètre clé de toute communication interpersonnelle, dans la mesure où il permet de discriminer finement les communications réussies, parmi toutes celles qui sont comprises. Un second paramètre orthogonal au premier, défini comme le contexte du locuteur, s'avère déterminant lorsqu'il s'agit d'aborder la réalisation concrète des communications interpersonnelles réussies. La déclinaison de ces deux paramètres dans le cadre de la messagerie électronique nous conduit à concevoir la notion de correspondance. Cette dernière constitue une généralisation du courrier électronique dont la mise en œuvre offre une condition suffisante de qualification des échanges réussis, via ce média. Une correspondance permet de prendre en compte pour chaque message, l'intention de communication et le contexte de son émetteur. Sa mise en œuvre impose l'application en certains points du réseau, de politiques spécifiques au domaine administratif de référence, qui prennent en argument l'intention de communication et le contexte courant de l'émetteur. Un second bénéfice apporté par ce concept concerne le niveau de personnalisation du service de messagerie qui atteint une granularité de finesse maximale, du fait qu'il peut s'appliquer de façon différenciée, à chaque occurrence de message. Ces travaux ont abouti à la description d'une architecture représentative accompagnée de la définition de trois extensions de standards existants (SUBMISSION, IMF et S/MIME). Notre approche a été illustrée à travers deux cas d'usages importants, conformes à des spécifications recommandées pour les domaines administratif (RGS- référentiel général de sécurité) et militaire (MMHS - Military Message Handling System).

ABSTRACT

The ease of use and efficiency of the email service contributed to its widespread adoption. It became an essential service and authorizing multiples and various uses (private, professional, administrative, governmental, military ...). However, all existing systems are technically reduced to the implementation of global policies, compiling in a static way a limited set of features. These approaches prevent differentiated adaptations of the system to the uses. The rigid and monolithic nature of these policies can moreover lead to unnecessary execution of expensive treatments or to the inability to simultaneously satisfy conflicting requirements.

We address this problem of the evolution of e-mail in the general context of interpersonal communication of a sender to a receiver. We identify the sender's intention of communication, as a key parameter of any interpersonal communication, insofar as it allows to finely discriminate the successful communications, between all the ones that are understood. A second parameter which is orthogonal to the first, defined as the context of the sender, is important because it allows to determine the successful aspect of an interpersonal communication. The declination of these two parameters in the electronic mail led us to define the concept of electronic correspondence. This one is a generalization of the email the implementation of which provides a sufficient condition of qualification successful exchanges via this medium. A correspondence allows taking into account for each message, the intention of communication and context of its sender. Its implementation requires in certain points of the network, the enforcement of specific policies depending of an administrative domain and which take as argument the intention of communication and the current context of the sender. A second benefit provided by this concept concerns the level of customization of messaging reaching a maximum granularity, because it can be applied in a differentiated way, to each message instance. These works led to the description of a representative architecture and the definition of three extensions to existing standards (SUBMISSION, IMF and S/MIME). Our approach has been illustrated through two main use cases, compliant with recommended specifications for administration (RGS - Référentiel Général de Sécurité) and military (MMHS - Military Message Handling System) domains.

REMERCIEMENTS

En premier lieu, je tiens à remercier mon directeur de thèse, Jean-Marie Bonnin, pour la confiance qu'il m'a accordée en acceptant d'encadrer ce travail doctoral, pour ses conseils et pour sa grande disponibilité.

J'exprime tous mes remerciements à Maryline Laurent et François Charoy, d'avoir accepté d'être rapporteurs, à Francine Krief, présidente du jury ainsi qu'à Benoit Martin, Charles Préaux, Philippe Letellier et Damien Roque pour leur participation à ce jury.

Je remercie également Olivier Lesbres, Directeur de DGA Maîtrise de l'information, de m'avoir permis de réaliser ces travaux de thèse à temps partiel. J'adresse toute ma gratitude à tous mes collègues et à toutes les personnes qui m'ont aidé dans la réalisation de ce travail.

Je tiens à remercier Philippe Letellier, pour avoir accepté de soutenir notre démarche de développement d'un prototype et d'avoir contribué à son financement.

Enfin, je remercie tout particulièrement Ahmed Bouabdallah, pour son accompagnement, sa disponibilité, son soutien et ses conseils avisés qui m'ont permis d'avancer et d'aboutir à ces travaux.

A Laura, Astrid et Aymeric.

PUBLICATIONS

Brevet

«*Architecture de soumission différenciée*», Brevet dépôt n° 1300752, 2014
L. Cailleux, A. Bouabdallah et S. Gombault. Extension PCT en cours

Publications

« *A New Security Service for Future Military Messaging* », Laurent Cailleux, MCC 2013 Military Communications and Informations System Conference, IEEE, 7-9 oc. 2013, Saint-Malo, France

« *A Confident Email System based on a New Correspondence Model* », Laurent Cailleux, Ahmed Bouabdallah et Jean-Marie Bonnin. ICACT 2014, the 16th international conference on advanced communications technology, IEEE, 16-19 feb. 2014, Pyeong-Chang, Korea

« *Building a Confident Advanced Email System using a new Correspondence Model* », Laurent Cailleux, Ahmed Bouabdallah et Jean-Marie Bonnin. Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference, IEEE, 13-16 may 2014, Victoria, Canada

« *Securing header fields with S/MIME* », Laurent Cailleux et Chris Bonatti. IETF, Internet draft, 26 july 2014, work in progress

«*A Policy-based Correspondence Model to Personalize Email Systems* », Laurent Cailleux, Ahmed Bouabdallah et Jean-Marie Bonnin. International Journal of Grid and Utility Computing (en cours de soumission)

Conférences sans actes

« *Trustedbird - Mozilla Thunderbird* », Laurent Cailleux, fOSSa 2010, January 2010, Grenoble, France

« *Trustedbird, un client de messagerie de confiance* », Laurent Cailleux, JRES 2009, November 2009, Nantes, France

PROTOTYPES

TrustedBird project. Client de messagerie open source basé sur Mozilla – Thunderbird mettant en œuvre les mécanismes des champs d'entête sécurisés.
http://adullact.net/plugins/mediawiki/wiki/milimail/index.php/Trustedbird_Project/fr

Projet AUDIENCE. Développement d'un serveur de soumission de message intégrant l'extension CORRESPONDENCE et basé sur le brevet cité ci-dessus.

Chapitre 1

Introduction

Le service de courrier électronique en raison de sa simplicité d'usage combinée à son efficacité, a constitué l'un des principaux vecteurs de popularisation d'Internet. Ce service a été progressivement introduit dans les environnements professionnels en se substituant aux services du fax et du courrier postal. Les différentes fonctionnalités offertes par le courrier électronique tels que l'envoi de documents électroniques, la rapidité d'acheminement sans la contrainte géographique, la possibilité d'envoi d'un message à des destinataires multiples représentent quelques atouts importants par rapport aux limitations du courrier postal. Les symbioses immédiates avec d'autres services électroniques tels que les calendriers, les carnets d'adresses, les gestionnaires de tâches, les points de partage de documents, les réseaux sociaux ... ont conduit à l'intégration de ce service dans les solutions de messageries collaboratives. Ce succès masque toutefois d'importantes lacunes. La messagerie électronique a été adoptée sans réelle conscience des risques, des inconvénients et souvent dans la méconnaissance des lois.

Le succès croissant de la messagerie électronique a également été accompagné d'une forte évolution des menaces. Il est important de noter que ce service a initialement été conçu sans prendre en compte les aspects de sécurité. Ce manque a été partiellement comblé dans le courant des années 2000 avec l'apport de services de sécurité mais ceci ne répond pas encore totalement aux exigences des particuliers et des entreprises. Les entreprises, plus ou moins conscientes de ces risques, ont dû s'adapter à ce contexte et ont déployé des solutions de sécurité répondant à des menaces particulières. Il est d'ailleurs intéressant de noter que 75% des informations critiques d'une entreprise se retrouvent dans les courriels[1]. Certaines entreprises et organisations ont contribué à la définition de guides, de recommandations et de standards visant à proposer des solutions pour les futurs systèmes de messagerie électronique, avec des succès plus ou moins reconnus. Le NIST[2], agence du Département du Commerce des États-Unis dont la mission est de promouvoir l'économie en développant notamment des standards, a publié une recommandation appelée *Guidelines on Electronic Mail Security* [1], dont l'objectif est de recommander les pratiques à respecter pour concevoir et mettre en œuvre des systèmes de messagerie sécurisée. L'IETF[3], organisation en charge du développement et de la promotion des standards de l'Internet, a publié de nombreux standards directement liés ou ayant trait aux concepts de messagerie sécurisée.

---

[1] http://www.ccsoftware.ca/email-archive/docs/email-archiving-advantages.pdf

[2] Le NIST (National Institute of Standards and Technology) est une agence du Département du Commerce des États-Unis. Son but est de promouvoir l'économie en développant des technologies, la métrologie et des standards de concert avec l'industrie. Cette agence a pris la suite en 1988 du National Bureau of Standards.

[3] L'IETF (Internet Engineering Task Force) est une organisation ouverte qui participe au développement et à la promotion des standards de l'Internet.

Le domaine militaire participe aussi, de son côté, activement à des travaux d'élaboration de normalisation des concepts de systèmes de messagerie. L'OTAN[4]a publié un document [2] qui est un recueil des exigences à respecter pour un système de messagerie opérationnel et qui définit trois niveaux de services (*high, medium et basic grades*). A chaque service correspond une liste d'exigences plus ou moins contraignantes. Le service de messagerie *high grade*, qui se rapproche des concepts de messagerie de confiance, définit les mécanismes à mettre en œuvre pour répondre à un besoin d'échanges critiques. En ce qui concerne les services *medium* et *basic grades*, ils couvrent le périmètre des échanges officiels, professionnels voire privés.

L'ensemble de ces initiatives a contribué à l'évolution des systèmes de messagerie tels que nous les connaissons aujourd'hui et notamment à leur sécurisation.

1.1 *Problématique*

Bien que contribuant à la réalisation de systèmes de messagerie sécurisée en fournissant des fonctionnalités avancées (analyse anti virale, filtrage de pourriels, authentification, intégrité, confidentialité, non répudiation, autorisation), les solutions abordées précédemment ne couvrent cependant qu'une partie du périmètre. L'utilisation du service de messagerie peut prendre différentes formes. Elle peut revêtir un caractère privé, cas par exemple des échanges de messages entre amis ou bien avec son médecin ou une administration. Elle peut également concerner des aspects professionnels, imposant en général des contraintes et des exigences particulières. L'utilisation des systèmes de messagerie s'inscrit donc dans un usage particulier : privé, professionnel, administratif, officiel, militaire...

Chaque type d'usage est régi par un contrat conforme à une politique, elle-même définie par le fournisseur du système de messagerie électronique. L'utilisateur aura confiance dans un système de messagerie qui comporte des services en adéquation avec l'usage qu'il fait du système. Ainsi, l'usage privé dans sa forme la plus simple nécessite des services de base. A contrario, l'usage militaire impose la mise en œuvre de nombreux services [2]. Pour chaque usage, le fournisseur du système de messagerie doit garantir à l'utilisateur, les services dédiés qui doivent s'inscrire dans le cadre d'une règlementation associée à l'usage. D'ailleurs, l'usage de la messagerie électronique est également régi par un cadre juridique. Ce dernier impose des règles que les fournisseurs de systèmes de messagerie doivent respecter en les intégrant dans leurs offres. En fonction de l'usage, le cadre juridique a un impact sur les fonctionnalités à mettre en œuvre.

Cette problématique d'usage de la messagerie peut être abordée avec une approche plus globale, sous l'angle de la théorie de la communication. Nous pouvons définir la communication comme une action permettant de communiquer, d'établir une relation avec autrui, de transmettre une information à un ou plusieurs autres individus. Cette communication pouvant prendre plusieurs formes (verbale, non verbale...). Dans sa forme la plus courante, la messagerie électronique propose un mode de communication interpersonnelle permettant l'échange d'informations entre plusieurs individus. Nous retrouvons ainsi les constituant génériques d'une communication interpersonnelle[3], à savoir un locuteur qui, dans le cas présent, correspond à l'auteur (émetteur) du message, un ou plusieurs interlocuteurs prenant la forme des destinataires

[4] L'OTAN (Organisation du traité de l'Atlantique Nord) est une organisation politico-militaire qui rassemble de nombreux pays occidentaux, dont le but premier est d'assurer leur défense commune contre les menaces extérieures ainsi que la stabilité des continents ou sous-continents européen et nord-américain.

du message, le message échangé et le système permettant sa transmission. Cependant, le service de messagerie électronique comporte un certain nombre de particularités liées à son principe de transmission asynchrone (pas de proximité lors de la communication, limitation des aspects conversationnels, non instantanéité des échanges...).

L'efficacité d'une communication interpersonnelle est dépendante de certains facteurs comme la qualité du moyen de transmission de l'information et le langage utilisé pour communiquer. Nous retrouvons ces aspects dans la messagerie électronique, avec le système de messagerie qui permet la transmission du message de son émetteur jusqu'à son destinataire et le format du message qui garantit une interopérabilité entre les utilisateurs. Cependant, le processus de communication interpersonnelle est également emprunt d'une caractéristique particulière appelée intention de communication. J. Searle [4] souligne que l'intentionnalité communicative est un préalable à tout acte de communication et que cette intention de communication aura un effet sur l'acte. L'adaptation de ces concepts à la messagerie électronique permettrait de distinguer les usages particuliers qui peuvent être faits du système. On classera parmi ces usages, les communications privées, professionnelles, administratives, opérationnelles... En corrélant un message avec l'usage qu'en fait son auteur, l'intention de communication fournit la bonne abstraction pour caractériser de manière différente chaque occurrence des messages.

Les différents cas d'utilisation pourraient être efficacement dérivés de l'intention de communication et de son association avec le contexte. L'évolution des technologies et notamment des terminaux a entrainé une modification dans les habitudes d'utilisation de la messagerie électronique. Alors que les utilisateurs accédaient, il y encore quelques années, à leur messagerie électronique à partir d'un poste informatique non mobile, ils n'hésitent plus aujourd'hui à utiliser leur smartphone[5] pour accéder à leur boite aux lettres, ceci à partir de différents réseaux publics (hôtels, gares, aéroports...) souvent "non maîtrisés". Cette pratique qui consiste à utiliser ses équipements personnels (téléphone, ordinateur portable, tablette électronique) dans un contexte professionnel, a connu un formidable essor et a été désigné sous l'acronyme de BYOD[6]. Cependant, cette pratique pose des questions relatives aux aspects juridiques, à la sécurité de l'information et à la protection des données. L'envoi d'un message confidentiel à partir d'un smartphone connecté à un réseau public devrait nécessiter la mise en œuvre de services de sécurité appropriés. Ces derniers pourraient être différents de ceux appliqués lorsqu'un même utilisateur procède à l'envoi d'un même type de message à partir de son poste informatique connecté au réseau de son entreprise. Le contexte (la localisation, le réseau d'accès, le type de terminal, l'heure...) dans lequel se trouve l'utilisateur apparait comme une nouvelle dimension qui doit être prise en compte. Le contexte est clairement orthogonal à l'intention de communication puisque pour une même intention, les services à mettre en œuvre seront dépendants du contexte.

Les systèmes de messagerie actuels proposent un ensemble de fonctionnalités pour tous ses utilisateurs. Dans certains cas, il est possible de définir des politiques pour des utilisateurs particuliers ou pour des groupes d'utilisateurs. Cependant, la description de politiques dédiées n'est pas standard et de tels systèmes pourraient être difficiles à mettre en œuvre. Pour cette raison, la plupart des systèmes de messagerie fournissent une politique monolithique applicable

[5] Téléphone portable qui assure des fonctions informatiques et multimédias (définition Petit Robert)..

[6] BYOD (abréviation de l'anglais « Bring your own device ») est une pratique qui consiste à utiliser ses équipements personnels dans un contexte professionnel. Cette pratique pose des questions relatives à la sécurité de l'information et à la protection des données, ainsi que sociales et juridiques. Wikipedia

pour tous les utilisateurs et quel que soit l'usage qui est fait du système. Ces solutions même si elles semblent efficaces, sont trop limitées, car elles ne laissent pas de place aux concepts d'usage et de contexte. Un usage sophistiqué spécifié par une intention particulière, peut seulement être supporté par un système de messagerie qui intègre un ensemble dédié de fonctionnalités. De plus, l'ensemble des fonctionnalités imposées pour deux intentions différentes pourrait être totalement différents. Les cas d'usages privé et professionnel sont, dans ce sens, intéressants puisqu'ils peuvent exiger des fonctionnalités opposées (respect de la vie privée vs archivage des messages).

Les fournisseurs de systèmes de messagerie font face à plusieurs défis. Ils doivent tout d'abord, proposer des solutions permettant de répondre à l'évolution des menaces. Ils doivent également mettre à disposition des utilisateurs, des services prenant en compte les usages et le contexte. Actuellement, ces fournisseurs proposent des systèmes dédiés à un usage précis prenant rarement en compte, et de manière limitée, le contexte courant, ce qui aboutit à la définition et l'application de politiques monolithiques. La nouvelle approche proposée dans cette thèse permettrait à des fournisseurs de services de messagerie, de proposer des offres de services encapsulant un ensemble de politiques adaptées aux différents besoins de leurs utilisateurs.

1.2 *Objectif de la thèse*

L'objectif de cette thèse est d'étudier et de définir précisément les services permettant de répondre aux limitations précédemment exposées et contribuant à la description de nouvelles solutions de messagerie avancée. Ces services doivent permettre l'application de politiques appropriées en fonction de l'usage et du contexte courant dans lequel se trouve l'utilisateur. Une des pistes d'amélioration serait de disposer d'un système de messagerie dont le socle serait capable de faire évoluer dynamiquement son niveau de protection et d'apporter des réponses aux différents usages du système.

Les conditions de réalisation de cette thèse imposent de proposer des concepts contribuant à la définition de futurs systèmes de messagerie civile et militaire. Pour cela, nous nous appuierons sur les spécifications du référentiel général de sécurité appelé RGS [5] dont le but est de fixer les règles que doivent respecter certaines fonctions contribuant à la sécurité de l'information (lors de l'échange d'information entre autorités administratives...). Nous prendrons également en compte les différentes spécifications existantes pour la définition d'un système de messagerie militaire, désigné sous l'acronyme MMHS[7], compatible avec les exigences de l'OTAN.

Les travaux de cette thèse doivent aboutir à la description d'une architecture représentative qui prend en compte les spécifications précédemment citées ainsi qu'à une description d'une potentielle mise en œuvre.

1.3 *Contributions*

Les travaux de cette thèse ont fait l'objet de plusieurs contributions.

[7] Military Message Handling System. MMHS est une spécification militaire définie dans l'ACP 123 [185] ainsi que dans le STANAG 4406 [111]. Cette spécification décrit des extensions à la norme X.400 (1992) [154].

Définition du concept de correspondance électronique

Afin de répondre aux limitations des systèmes de messagerie actuels et après une étude de ce domaine complexe, nous avons défini un nouveau concept, appelé correspondance électronique. Ce concept s'inspire de la forme basique de communication interpersonnelle. Lorsque deux ou plusieurs personnes communiquent, des messages sont envoyés et reçus. Une communication interpersonnelle est considérée comme réussie quand l'émetteur et les destinataires comprennent le message et l'intention associée. L'application de ces concepts dans un système de messagerie asynchrone impose une analyse détaillée du fonctionnement d'un tel service, ce qui nous a permis de décrire le principe de cycle de vie des messages électroniques. En effet, la communication dans ce cas, n'est pas directe, mais fait l'objet d'un transport à travers un système pouvant être composé de nombreux équipements. L'étude de la communication interpersonnelle nous a également permis de définir la notion d'intention de communication, préalable à tout échange. Nous avons associé cette notion d'intention, à l'usage qu'un utilisateur peut faire du système de messagerie. Par exemple, une intention de communication privée sera associée à un type d'usage. Les échanges professionnels, pour leurs parts, peuvent se décliner dans des usages plus élaborés. Nos travaux ont également abordé les aspects liés au contexte. Une communication entre individus pourra prendre plusieurs formes en fonction du contexte. La localisation, le type de terminal, le réseau d'accès ou encore l'heure, sont autant de critères pouvant influer sur la communication.

En se basant sur ces travaux, nous avons défini un modèle de correspondance électronique qui permet d'appréhender le concept de système de messagerie d'une manière plus globale en envisageant l'application de politiques adaptées aux usages qui sont faits du système et en fonction du contexte dans lequel se trouve l'utilisateur.

Architecture basée sur le modèle de correspondance

Dans ces travaux de thèse, nous avons défini une architecture représentative basée sur le modèle de correspondance électronique. Celle-ci permet, tout en garantissant l'interopérabilité nominale des systèmes de messagerie, d'étendre les principes d'architectures de messagerie de l'Internet proposé par l'IETF [6].

Extension de standards existants

Trois extensions aux standards actuels du service de messagerie ont été définies lors des travaux de cette thèse. La figure suivante propose un empilement protocolaire et le positionnement de ces extensions. Il est important de noter que ces trois extensions proposent de nouvelles fonctionnalités sans remettre en cause le fonctionnement initial des protocoles étendus.

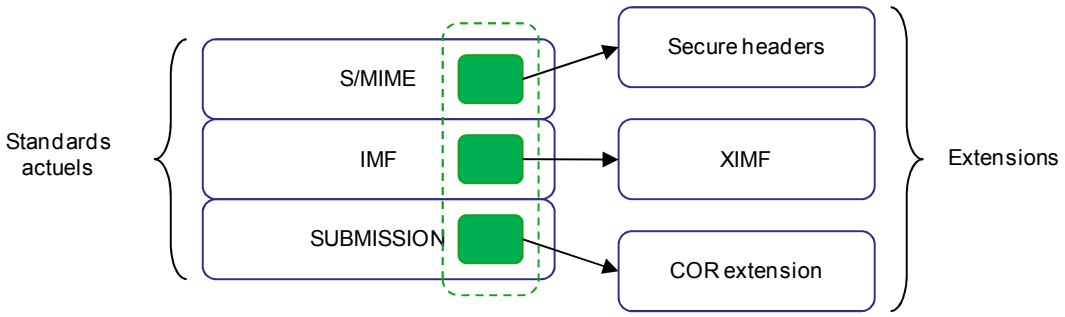


Figure 1 : Localisation des extensions décrites dans la présente thèse

- *Extension du standard de soumission de message*

  Les systèmes de messagerie actuels sont basés sur des standards simples, ce qui a contribué à leurs formidables développements. Cependant, cette simplicité impose aussi une limitation des usages que les utilisateurs peuvent faire du système. L'offre proposée par un fournisseur est souvent associée à un type d'usage et à la mise en œuvre d'un système dédié. Afin d'introduire la notion de système de messagerie multi-usages, nous avons décrit une extension du standard SUBMISSION[7], basé sur le standard SMTP [8] (Simple Mail Transfer Protocol). Cette extension permet une négociation de l'usage lors de la transaction du message.

  Cette extension a été intégrée dans un prototype développé dans le cadre du projet AUDIENCE[8].

  L'objectif est d'initier auprès de l'IETF, un processus de standardisation de cette extension, décrite à ce jour dans un document de travail.

- *Extension du standard de format de message*

  Lors de l'acheminement du message, le transport des informations liées à l'usage est nécessaire. Sans cela, le système ne peut détecter l'usage et appliquer les traitements idoines. Le standard de transmission de messages, à savoir SMTP [8], propose des mécanismes d'extension. L'idée d'une extension du standard SMTP semble au premier abord séduisante. Cependant, le principe de fonctionnement du protocole SMTP ne permet pas de garantir le transport des extensions tout au long de la chaîne de transmission du message. Une solution alternative est de se baser sur le standard de format de message IMF [9] (Internet Message Format) et de l'étendre. Dans cette thèse, nous avons décrit une extension qui permet l'ajout d'informations liées à l'usage dans la partie entête des messages. Ces informations peuvent ensuite être transportée de l'émetteur du message jusqu'à son destinataire final.

  Cette extension a été intégrée dans un prototype développé dans le cadre du projet Trustedbird[9].

[8] AUDIENCE (Architecture de soUmission DIfferENciee de Courrier Electronique) est un projet de l'Institut Mines-Telecom.

[9] Le client Trustedbird a été développé dans le cadre d'un projet piloté par la DGA (Direction Générale de l'Armement). L'objectif initial de ce client était de fournir des services de sécurité étendus.

L'objectif est d'initier auprès de l'IETF, un processus de standardisation de cette extension, décrite à ce jour dans un document de travail.

- *Extension du standard de sécurisation de message*

  La sécurisation des échanges de messages est devenue un défi auquel doivent faire face les organisations utilisatrices. Les deux principales solutions de sécurisation actuelles sont basées sur les standards TLS [10] (Transport Layer Security) et S/MIME [11] (Secure Multipurpose Internet Mail Extensions). Le premier décrit des mécanismes de sécurisation de la transmission d'un message entre un composant client et un composant serveur. Il ne peut être utilisé pour garantir une sécurisation du message lors de son acheminement (de l'émetteur initial jusqu'au destinataire final) au sein du système de messagerie. Le second standard propose des mécanismes de sécurisation du message. Cela permet le transport d'un message sécurisé, quel que soit le système traversé. Cependant, les mécanismes décrits dans le standard S/MIME ne s'appliquent que sur le corps du message et non sur la totalité du message (entête et contenu). Le principe d'extension du format de message que nous avons décrit précédemment impose l'ajout de métadonnées dans la partie entête du message. Afin de garantir la sécurisation de ces données, nous avons décrit une extension du standard S/MIME. Grâce à cette extension, il est possible de sécuriser la totalité du message.

  Un processus de soumission[10] de cette extension du standard S/MIME a été initié auprès de l'IETF. Un accord de publication du document avec un statut expérimental a été transmis par l'IETF.

Prototypes

Deux prototypes ont été développés dans le cadre de cette thèse.

Un premier développement a concerné un client de messagerie, appelé *TrustedBird*. Ce client, basé sur *Mozilla Thunderbird*[11], intègre un composant embarquant des politiques de correspondances et un composant permettant leurs applications. Ce client de messagerie supporte également l'extension CORRESPONDENCE et l'extension de format de messages étendus.

Le second développement, réalisé dans le cadre d'un projet nommé AUDIENCE, a concerné un serveur de soumission. Ce serveur intègre l'extension CORRESPONDENCE et permet de s'interfacer avec une architecture d'autorisation. Ces travaux ont fait l'objet d'un dépôt de brevet.

[10] https://datatracker.ietf.org/doc/draft-cailleux-secure-headers/

[11] Le client Mozilla Thunderbird est un client de messagerie libre distribué gratuitement par la Fondation Mozilla et issu du Projet Mozilla.

1.4 *Plan du rapport*

Ce rapport, composé de 9 chapitres et de 4 annexes, est organisé de la manière suivante. Le premier chapitre constitue l'introduction. Les chapitres 2, 3, 4 et 5 proposent un état de l'art. Les chapitres 6, 7 et 8 traitent de notre contribution pour résoudre le problème posé. Enfin, le chapitre 9 propose une conclusion et présente les perspectives futures. Le détail du travail dans les différents chapitres est organisé comme suit :

Le chapitre 1 est une introduction au problème étudié dans le cadre de cette thèse.

Les chapitres 2, 3, 4 et 5 présentent l'état de l'art. La définition d'un modèle qui réponde aux exigences et limitations décrites dans le chapitre d'introduction a nécessité une étude qui couvre plusieurs domaines : les concepts autour de la messagerie électronique, les aspects liés à la sécurisation de ce service, le contrôle de la messagerie par l'application de politiques et l'intention de communication dans les systèmes de messagerie.

Le chapitre 6 détaille le concept de correspondance électronique. Ce chapitre introduit également la notion de cycle de vie du message et décrit les évènements en charge de son acheminement. Le concept de politiques de correspondances est également présenté dans ce chapitre.

Le chapitre 7 propose une description de deux exemples de politiques de correspondance dans des systèmes représentatifs. Ces deux exemples sont basés sur le RGS et sur les différentes spécifications existantes pour la définition d'un système de messagerie militaire de type MMHS.

Le chapitre 8 présente une architecture détaillée mettant en œuvre les concepts de correspondance électronique ainsi que les évolutions et modifications à apporter dans les différents composants du système.

Le chapitre 9 propose une conclusion et les nouvelles perspectives ouvertes par les travaux de cette thèse.

L'annexe 1 propose la description d'un cycle de vie complet d'un message électronique.

L'annexe 2 comporte le document "Securing header fields with S/MIME" soumis à l'IETF.

L'annexe 3 comporte le brevet, déposé dans le cadre de cette thèse.

L'annexe 4 propose une première étape de formalisation du modèle de correspondance développée à l'aide du langage formel IOA (*Input/Output Automata*) et de l'outil Tempo[12].

[12] http://www.veromodo.com/code/plugins.html

Partie 1

Etat de l'art

Chapitre 2

La messagerie électronique

2.1 ***Origine de la messagerie électronique***

Avant de voir en détail les concepts de la messagerie électronique, il est intéressant de faire un court historique de l'évolution de ce service. La messagerie électronique est un service qui est étroitement lié à l'histoire de l'Internet. Ce service a contribué, avec l'émergence de la technologie Web, au formidable développement et au succès du réseau Internet. Dès l'apparition des premiers programmes informatiques, les ingénieurs ont voulu développer des solutions permettant l'échange d'informations entre ordinateurs. Dans les années 60, les premiers systèmes de messagerie ont vu le jour. Le *Time-Shared Operating System* intégrait un système de remise de message. En 1971, D. Watson publie un premier document de description de la messagerie électronique, le RFC196[13] [12], appelée "*Mail Box Protocol*". L'idée était de fournir un mécanisme permettant au *Network Information Center*[14] de distribuer des documents aux différents sites du réseau Arpanet[15]. En 1972, le premier protocole de transfert de message est publié et les premiers outils de création et de gestion de messages sont développés. Ces premiers travaux sont les prémices d'une évolution qui ne s'est pas arrêtée lors des quarante années qui ont suivies [13] pour arriver aux systèmes de messagerie que nous connaissons aujourd'hui.

2.2 ***Concepts de messagerie électronique***

L'objectif principal d'un système de messagerie électronique est de permettre l'acheminement d'un message émis par un utilisateur ou une application à destination d'un utilisateur ou d'un système. Trois aspects fondamentaux sont présents dans un système de messagerie électronique: les agents, le message et son acheminement.

La particularité de la messagerie électronique par rapport notamment à la messagerie instantanée est liée à son caractère asynchrone. Cet asynchronisme permet à l'expéditeur d'envoyer un message et à son destinataire de lire le message à des moments différents. Les systèmes asynchrones imposent que les messages soient stockés sur un ordinateur.

---

[13] Les RFC (Requests for Comments) sont des documents techniques ou des notes organisationnelles dont le sujet est Internet. Ils couvrent de nombreux aspects (concepts, protocoles, procédures, opinions...).

[14] Network information center (NIC) était autrefois le nom donné aux organismes qui géraient les réseaux informatiques, notamment Internet, en assurant l'enregistrement des paramètres uniques (les adresses, par exemple) et en ayant un rôle de conseil et d'information, d'où leur nom.

[15] ARPANET ou Arpanet (acronyme anglais de « Advanced Research Projects Agency Network ») est le premier réseau à transfert de paquets développé aux États-Unis par la DARPA. Le projet fut lancé en 1969 et la première démonstration officielle date d'octobre 1972.

2.3 *Architecture d'un système de messagerie électronique*

Les systèmes de messagerie électronique actuels ont été développés en se basant sur les standards SMTP [8], IMF [9], MIME (standard composé de cinq documents [14][15][16][17][18]). Ce principe simple a permis une adoption rapide qui a ensuite entrainé un formidable développement de ces systèmes. Toutefois, ce succès a également entrainé le développement de systèmes qui ont fait face à des problèmes d'interopérabilité. En 2009, afin de remédier à ces développements pouvant comporter des incohérences souvent dues à des incompréhensions, un document permettant de clarifier les architectures de messagerie a été publié par l'IETF, le RFC 5598 [6]. Il présente les grands principes d'architectures de messagerie électronique basées sur les protocoles de l'Internet. Ce document propose une description globale de l'ensemble des composants, services et utilisateurs regroupés au sein d'une architecture appelée IMA (Internet Mail Architecture). Cette architecture permet d'effectuer des échanges dits de « bout en bout ». Cette terminologie correspond à l'envoi du message et à l'ensemble des remises qui résultent de son transit au sein d'un MHS (Mail Handling System). L'acheminement d'un message est réalisé par une infrastructure composée :

- d'un objet message appelé également message,
- d'un espace d'adressage global,
- d'un mécanisme de transfert asynchrone de point à point

Il est à noter qu'un système compatible avec la spécification IMA n'intègre pas de mécanisme de transfert différencié des messages. Les politiques appliquées sur les messages ne tiennent pas compte de l'importance des informations transportées, de leurs niveaux de sensibilité... Cependant, le transfert de messages avec traitement des priorités a fait récemment l'objet de travaux et une extension du protocole SMTP a été publiée[19]. De plus, les utilisateurs d'un tel système ne doivent pas nécessairement être présents au moment de l'échange du message. Ce principe d'asynchronisme a également contribué au succès de la messagerie électronique.

IMA propose une description des rôles présents dans une architecture de messagerie. Ces rôles sont présentés sous la forme de trois types d'acteurs.

- *User*
- *Message Handling Service* (MHS)
- *Administrative Management Domain* (ADMD)

L'acteur *User* peut prendre plusieurs formes. Il peut correspondre à l'auteur du message (*Author*). Dans ce cas, il est en charge de la création du message, de son contenu et de la liste des destinataires. L'acteur *User* peut également être associé aux destinataires du message (*Recipient*). Il est à ce titre consommateur du message remis. Enfin, les deux autres acteurs sont des formes particulières du rôle User. Le premier appelé *Return Handler* est en charge du service des notifications qui peuvent être générées dans le cas de dysfonctionnements ou pour informer l'auteur du message que ce dernier a été correctement remis. Le dernier acteur est le médiateur (*Mediator*) dont le rôle est plus complexe. Il peut recevoir, agréger, reformuler et redistribuer les messages entre l'auteur du message et le destinataire. Le cas classique de médiateur est la liste de

diffusion qui offre un mécanisme d'expansion de liste et permet de dédier à ce rôle la création des messages et leurs transmissions à tous les destinataires présents dans la liste associée.

L'acteur MHS est en charge de la transmission du message de l'auteur jusqu'aux destinataires. Pour cela, il génère, visualise ou modifie les données nécessaires au transfert du message. Il ne devrait pas modifier le message. Pour réaliser la transmission du message, le MHS s'appuie sur quatre rôles. Le premier rôle est *Originator* qui garantit la validité du message par rapport aux standards et aux politiques locales. Le rôle *Author* créé le message alors que le rôle *Originator* permet sa manipulation et sa transmission. Le rôle *Receiver* réalise la remise finale. Dans certains cas, cette remise pourrait être redirigée vers une autre adresse. L'acteur MHS comporte un rôle *Relay* qui effectue le routage des messages à destination des rôles *Recipient*. Le rôle *Relay* ne doit pas modifier le message dont il a la charge, sauf pour des cas particuliers comme le changement d'encodage du contenu. Cependant, afin d'éviter un bouclage dans le routage du message et assurer une traçabilité du message, le rôle Relay ajoute une information dans le message sous la forme d'un champ d'entête. Le dernier rôle de l'acteur MHS est *Gateway*. Ce rôle assure l'interconnexion entre des systèmes de messagerie hétérogènes.

Le dernier acteur est l'ADMD. Une ADMD est définie comme un acteur principal d'une architecture de messagerie et est associée avec un nom de domaine. Une ADMD est une entité indépendante qui est sous la responsabilité d'une autorité administrative. Elle applique, à ce titre, un ensemble indépendant de politiques. Dans les systèmes de messagerie, il existe deux principaux modèles. Dans le premier modèle, on parle d'intra-ADMD (ou intra-domaine) et dans le second d'inter-ADMD (ou inter-domaine). En intra-ADMD, les messages sont transmis entre composants présents dans un même environnement d'administration et utilisant le même nom de domaine. Dans ce type de modèle, l'application de politiques est simplifiée. Il y a une seule autorité administrative et tous les composants du système de messagerie sont sous la responsabilité de cette autorité. Dans le cas d'échanges dans un modèle inter-ADMD, les messages sont échangés entre différentes ADMD. L'émetteur et le destinataire du message n'appartiennent pas à la même ADMD et n'appliquent pas les mêmes politiques car chaque ADMD dispose d'un ensemble indépendant de politiques. Cependant, des relations de confiance et des arrangements complexes peuvent être établis entre ADMD, permettant ainsi l'application de politiques compréhensibles et acceptées par les utilisateurs ou composants des ADMD concernées. Une architecture globale pourrait intégrer plusieurs ADMD. C'est le cas typique d'Internet qui est composé d'une multitude d'ADMD.

Une liste récapitulative des différents rôles et acteurs associés est proposée ci-dessous.

- Users
    - Authors
    - Recipients
    - Return handler
    - Mediator
- MHS
    - Originator
    - Receiver
    - Relay
    - Gateway
- ADMD

Une architecture IMA est constituée de plusieurs types de composants appelés aussi agents. Le schéma décrit dans la Figure 2, extrait du RFC 5598 [6], illustre ces différents agents et les protocoles permettant leurs interactions dans une architecture représentative. Les agents présents dans un système de messagerie sont:

- le MUA (*Message User Agent*) interagit avec le système à la place de l'acteur *User*.
- le MSA (*Mail Submission Agent*) accepte le message soumis par le MUA et applique les politiques de l'ADMD hôte.
- le MTA (*Mail Transfer Agent*) est en charge de relayer le message jusqu'à sa destination. Un MTA intègre à la fois une partie cliente et une partie serveur. Le MTA ne doit pas modifier l'enveloppe ni le message, seul l'ajout de trace est réalisé. Il existe deux types de MTA. le *boundary* MTA (ou *Border* MTA) qui interagit avec des MTA d'autres ADMD et le *final* MTA qui transfert le message au MDA.
- le MDA (*Mail Delivery Agent*) est en charge de la remise du message dans la boite aux lettres du destinataire.
- le MS (*Message Store*) permet à un MUA de stocker les messages sur le long terme.

Dans la suite de ce rapport, nous parlerons d'agent pour faire référence à un composant de l'architecture IMA.

Le MUA est composé de :

- aMUA (*author MUA*) crée un message et réalise la soumission initiale dans l'infrastructure de messagerie via le MSA.
- rMUA (*recipient MUA*) interagit avec le système à la place du destinataire pour traiter les messages récupérés.

Le MSA est composé de :

- aMSA (*Author-focused MSA functions*)
- hMSA (*MHS-focused MSA functions*) prend la responsabilité du transit pour un message qui est conforme aux standards et aux politiques locales.

Le MDA est composé de :

- hMDA (*MHS-oriented MDA*) correspond à la partie serveur SMTP.
- rMDA (*Recipient-oriented MDA*) réalise les actions de remise que le destinataire a spécifié.

Le MS est composé de aMS (author MS) et de rMS (recipient MS) qui permettent d'archiver et d'organiser les messages de différentes manières.

L'entrée et la sortie d'un message dans un MHS sont accompagnés d'un transfert de responsabilité. Ces transferts de responsabilité sont réalisés dans le MSA pour la partie soumission et dans le MDA pour la partie remise du message. Ces deux transferts de responsabilité sont décrits dans la Figure 2 avec les transitions (S) pour soumission et (D) pour remise.

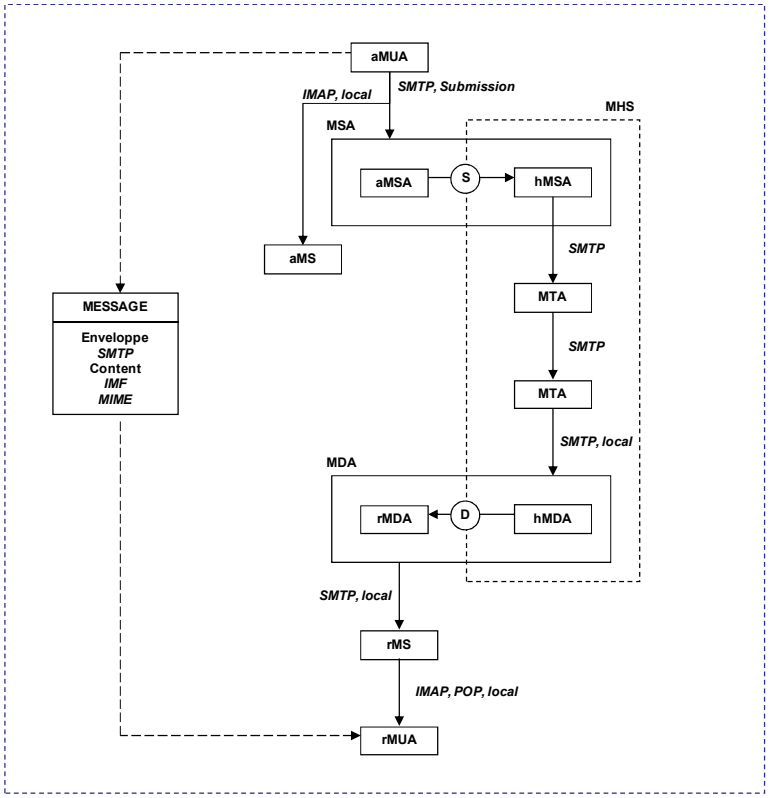
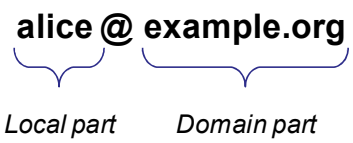


Figure 2 : Architecture globale d'un système de messagerie électronique

2.4 *L'adresse électronique*

La messagerie électronique est indissociable du concept d'adresse électronique. Une adresse est une chaîne de caractères qui désigne une boite aux lettres. Une adresse de messagerie est composée de trois éléments. Une partie gauche (*local part*) qui comporte en général le nom de l'utilisateur ou un identifiant s'y référençant. Une partie droite (*domain part*) qui contient le nom du domaine identifiant généralement l'organisation qui héberge la boite à lettres. Un caractère de séparation « @ ». Le nom de domaine consiste en un ou plusieurs composants séparés par des points. La description du format des adresses électroniques est fournie dans le RFC5322 [9] et le RFC3696[20] apporte une clarification sur les restrictions dans la syntaxe d'une adresse électronique.



Le RFC5321 [8] propose la description en ABNF d'une adresse de messagerie.

```
Mailbox = Local-part "@" ( Domain / address-literal )

Local-part     = Dot-string / Quoted-string                ; MAY be case-sensitive
```

```
address-literal  = "[" ( IPv4-address-literal / IPv6-address-literal / General-address-literal ) "]"

Domain         = sub-domain *("." sub-domain)

sub-domain     = Let-dig [Ldh-str]

Let-dig        = ALPHA / DIGIT
```

2.5 *Le message électronique*

Afin de permettre l'échange d'informations entre individus, il a été nécessaire de définir un format de message électronique compréhensible par les différents systèmes utilisateurs. Nous ne rentrerons pas dans les détails historiques de la messagerie électronique, mais ce qui est nécessaire de retenir est que la première étape de standardisation a été publiée dans le document RFC561 [21] en septembre 1973 [13]. Ce document nommé «*Standardizing Network Mail Headers* » proposait une première description du format de message électronique. Un message était un objet *mailtext* composé d'une partie entête (*header*) et d'une partie contenu (*message*). La partie entête était un ensemble de champs séparés par un retour à la ligne.

En 1982, différentes discussions ainsi que la transition vers le réseau Internet ont abouti à la publication d'une version majeure du standard de format de message électronique ; le RFC822 [22]. En plus d'une clarification de la description de champs déjà présents, les principales évolutions concernaient la description d'un champ *Received* dont l'objectif était de permettre le suivi du chemin du message à travers le système de messagerie électronique. Ce principe imposait aux serveurs en charge de la transmission du message d'interpréter le message afin d'enrichir son contenu, ce qui fut un changement notable dans le fonctionnement initial de la messagerie électronique. En effet, les serveurs n'interprétaient pas les messages qu'ils échangeaient. D'autres champs d'entête ont également vu le jour, notamment les champs relatifs au mécanisme de retransmission du message (*forwarding*). Le document RFC822 est considéré comme le standard de base du format de message électronique. Deux révisions de ce document ont été publiées. Le RFC2822 [23] (en 2001) et le RFC5322 [9] (en 2008) ont apporté quelques modifications qui portaient notamment sur l'interdiction d'intégrer le routage du message dans les adresses et sur la limitation de la longueur des lignes du message. Le titre de la dernière publication est « *Internet Message Format* ». Pour faire référence à cette description de format de message, nous utiliserons l'acronyme IMF.

A ce jour, le format du message IMF est composé de deux parties distinctes :

- l'entête (*header*) constitué d'une séquence de ligne de caractères avec une syntaxe spéciale définie dans le RFC5322
- et le corps (*body*) qui suit la partie entête et est constitué d'une séquence simple de caractères.

La séparation entre la partie entête et la partie corps du message est réalisée grâce à une ligne vide. La Figure 3 décrit le format d'un message IMF.

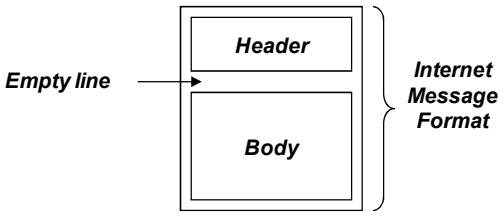


Figure 3 : structure d'un message IMF

Les champs d'entête sont des lignes structurées composées d'un nom de champ suivi par le caractère « : » puis par la valeur du champ et terminées par CRLF. Un nom de champ est composé de caractère US-ASCII (caractères compris entre les valeurs 33 et 126 de la description ASCII). La valeur d'un champ est également composée de caractères US-ASCII mais elle peut comporter des espaces, tabulations ou encore retour à la ligne. Ce dernier principe est appelé *folding*. Le RFC5322 décrit un certain nombre de champs d'entête standard, auxquels il associe un caractère obligatoire ou optionnel. Les principaux champs sont :

Champs	Description
From	Adresse de l'auteur du message
Sender	Adresse de l'émetteur du message
Reply-to	Adresse de retour
In-reply-to	Identifiant du message lors d'une réponse
To	Adresses des destinataires principaux
Cc	Adresses des destinataires en copie
Bcc	Adresses des destinataires en copie cachée
Message-ID	Identifiant unique du message
Subject	Objet du message
Orig-date	Date et heure à laquelle l'auteur a créé le message

Tableau 1: Principaux champs d'entête présents dans IMF

Il est intéressant de noter que seuls deux champs sont obligatoires : *orig-date* correspondant à la date et l'heure à laquelle le message est complet et prêt a être soumis au système de message et *from* qui comporte l'adresse de la boite aux lettres source du message. Le site de l'IANA[16] référence l'ensemble des champs d'entête ayant fait l'objet d'un document de standardisation[17].

L'annexe A du document RFC5322 propose un certain nombre d'exemple de messages respectant le format IMF. Voici un exemple d'un message tiré de ce document :

```
From: John Doe <jdoe@machine.example>
Sender: Michael Jones <mjones@machine.example>
To: Mary Smith <mary@example.net>
Subject: Saying Hello
Date: Fri, 21 Nov 1997 09:55:06 -0600
Message-ID: <1234@local.machine.example>
```

[16] L'Internet Assigned Numbers Authority (IANA) est une organisation dont le rôle est la gestion de l'espace d'adressage IP d'Internet, et des autres ressources partagées de numérotation requises soit par les protocoles de communication sur Internet, soit pour l'interconnexion de réseaux à Internet. (wikipedia)

[17] http://www.iana.org/assignments/message-headers/message-headers.xhtml

```
This is a message just to say hello.
So, "Hello".
```

Le standard IMF est limité aux caractères ASCII (*American Standard Code for Information Interchange*). Cette limitation ne permettait pas la transmission de caractères accentués et encore moins de pièces jointes. Afin de répondre à cette limitation et de permettre de créer des messages intégrant des données non textuelles comme des documents bureautiques, des images ou encore du son, le standard MIME (*Multipurpose Internet Mail Extensions*) a été spécifié.

Afin de permettre à IMF d'embarquer des données non textuelles sans introduire d'incompatibilités, MIME propose la description du type de contenu. Pour cela, il décrit:

- un champ d'entête *MIME-Version* pour indiquer à l'agent de traitement que le message est basé sur le standard MIME.
- Un champ d'entête *Content-Type* qui spécifie le type et le sous-type de données présentes.
- Un champ d'entête *Content-Transfer-Encoding* qui spécifie la transformation qui a été effectuée sur le corps du message
- Deux champs d'entête additionnels qui sont utilisés pour fournir des détails supplémentaires sur les données embarquées.

Le message peut être constitué de plusieurs parties (ou plusieurs pièces jointes). Pour cela, MIME propose un concept de séparateur entre les différentes parties. Le champ d'entête *Content-Type* dispose d'un attribut *boundary* qui définit le séparateur de partie. L'exemple ci-dessous correspond à un message IMF qui embarque deux parties texte et une pièce jointe au format PDF.

```
...
Message-ID: <16906767.479561214469730330.www@winf8215>
Subject: test
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="----=_Part_26626_29952974.1214469730310"

------=_Part_26626_29952974.1214469730310
Content-Type: multipart/alternative;    boundary="----=_Part_26627_2498572.1214469730310"

------=_Part_26627_2498572.1214469730310
Content-Type: text/plain; charset=UTF-8
Content-Transfer-Encoding: quoted-printable

texte 1

------=_Part_26627_2498572.1214469730310
Content-Type: text/html; charset=UTF-8
Content-Transfer-Encoding: quoted-printable

texte 2

------=_Part_26627_2498572.1214469730310--

------=_Part_26626_29952974.1214469730310
Content-Type: application/pdf; name="/home/Desktop/test.pdf"
content-transfer-encoding: base64
Content-Disposition: attachment; filename="/home/Desktop/test.pdf"

JVBERi0xLjQKJeLjz9MKMSAwIG9iajw8L1Byb2R1Y2VyKGh0bWxkb2MgMS44LjI3IENvcHlyaWdo
dCAxOTk3LTIwMDYgRWFzeSBTb2Z0d2FyZSBQcm9kdWN0cywgQWxsIFJpZ2h0cyBSZXNlcnZlZC4p
L0NyZWF0aW9uRGF0ZShEOjIwMDgwNjI1MTIzMDI2LTAyMDApPj5lbmRvYmoKMiAwIG9iajw8L1R5
....
```

MDYyYmJjZWQ3MzZmZmU3YTliMTJlNzQ2YzA2ZWFlZj48NTA2MmJiY2VkNzM2ZmZlN2E5YjEyZTc0
NmMwNmVhZWY+XT4+CnN0YXJ0eHJlZgoyMzMwNDgyCiUlRU9GCg==
------=_Part_26626_29952974.1214469730310--

MIME a été initialement décrit dans cinq documents : RFC2045 [14], RFC2046 [15], RFC2047 [16], RFC2048 [17] et RFC2049 [18]. La RFC2045 est complétée par la RFC2077. La RFC2048, devenue obsolète, est remplacée par les RFC4289 et RFC4288.

2.6 *Le standard SMTP*

Parallèlement à la définition d'un standard de format de message, il est devenu rapidement nécessaire de proposer un standard pour la transmission des messages sur le réseau Internet. Dans le courant des années 70, certains systèmes intégraient des solutions de messagerie électronique plus ou moins propriétaires. En 1973, une première solution a émergé. Le principe était basé sur la transmission d'un fichier à partir des mécanismes liés au protocole FTP (File Transfer Protocol). Le RFC458 proposait notamment la récupération de message à l'aide de deux nouvelles commandes ajoutées au protocole FTP. En 1980, V. Cerf et J. Postel ont rédigé le RFC771 dont l'objectif était de proposer un plan de transition des protocoles de messagerie utilisés sur le réseau Arpanet vers une nouvelle génération de protocoles de messagerie à utiliser sur le réseau Internet. Parallèlement, la même année le RFC 773 a apporté un complément technique à ce plan de transition et le RFC772 a proposé une description du protocole MTP (Mail Transfer Protocol) proposant des mécanismes d'interopérabilité entre les protocoles de messagerie des réseaux Arpanet et Internet.

En juillet 1981, J. Postel et S. Sluizer ont décrit un nouveau concept **d'enveloppe de message** dans deux documents. Dans le RFC784, ils ont mentionné qu'il était nécessaire de disposer pour chaque message de deux fichiers : Un premier permettant de contenir les informations nécessaires à la transmission et un second qui comporte le message. Dans le RFC785, ils ont décrit la structure interne du fichier d'enveloppe. Le protocole MTP étant critiqué pour sa complexité, un nouveau protocole a été défini en 1982 sous l'acronyme SMTP (Simple Mail Transfer Protocol). Ce nouveau protocole qui a fait l'objet d'une publication dans le RFC821 [24] est en soit plus simple que MTP car il propose des commandes avec zéro ou un seul argument contrairement à MTP. Il intègre également le nouveau concept d'enveloppe. Le protocole SMTP a ensuite subi une première révision en 2001 à travers le RFC2821[25]. Le retrait de certaines commandes et surtout l'interdiction du routage source explicite ont modifié le protocole. Ce dernier point est présenté en détails dans le chapitre décrivant le routage des messages basé sur les noms de domaines. En 2008, le RFC5321 [8] est venu compléter le standard SMTP.

Comme cela est précisé dans le standard SMTP, l'objectif de ce protocole est de transférer des messages de manière fiable et efficace. Le standard SMTP, décrit en détails dans [26], propose un mécanisme appelé « *store-and-forward* » (stockage du message puis transmission de celui-ci) basé sur un modèle « *push*». Dans ce modèle, l'agent qui dispose du message le transmet à l'agent suivant contrairement au modèle « *pull*». Une fonctionnalité importante de SMTP est sa capacité à transférer les messages entre de multiples réseaux via un mécanisme appelé « relayage de message». Ce mécanisme consiste à transmettre le message de serveur en serveur jusqu'à la remise du message au serveur hébergeant la boite aux lettres du destinataire. Grâce à ce

principe, le message peut transiter à travers différents réseaux avant d'atteindre le serveur cible. Ce principe de routage est basé sur la résolution du nom du prochain serveur. SMTP ne permet pas de récupérer les messages remis dans les boites aux lettres. Les protocoles POP3 et IMAP4 ont été définis pour cela. Ils sont présentés dans la suite de ce rapport.

Avant de décrire le modèle de base du standard SMTP, il convient de définir les termes connexion, session et transaction. Un schéma illustre ces notions dans la Figure 5.

Une connexion correspond au canal TCP qui est établi entre un client SMTP et un serveur SMTP. Afin de permettre l'établissement d'une connexion, le serveur SMTP dispose d'un processus qui est en écoute sur le port 25. Une connexion SMTP est terminée quand le client envoie une commande QUIT. Le serveur répond ensuite avec un code de réponse positif et ferme la connexion.

Une session SMTP est initiée quand un client ouvre une connexion avec un serveur SMTP et que le serveur répond avec un message d'ouverture. Une session se termine avec la commande QUIT. Il ne peut y avoir qu'une session par connexion.

Lorsque que la connexion est établie et que la négociation initiale est complète alors le client SMTP peut initier une transaction de message. Une transaction consiste en une série de commandes émises par le client et de réponses renvoyées par le serveur. Les commandes comportent des informations concernant l'émetteur, les destinataires et le contenu du message. Les réponses consistent en des codes de retour aux commandes. Une transaction débute avec la commande MAIL FROM et se termine avec la commande "." qui correspond à une séquence de fin de message. Une session peut être composée de zéro ou de plusieurs transactions qui s'enchainent successivement. Dans tous les cas, un client doit envoyer une commande HELO ou EHLO avant de débuter une transaction SMTP.

Le modèle basique de SMTP, illustré dans la Figure 4, est le suivant. Dès que le message est prêt à être transmis, un client SMTP établit une connexion avec le serveur SMTP. Le client SMTP détermine l'adresse de la machine hébergeant le serveur SMTP en résolvant le nom de domaine du serveur SMTP. Ce serveur peut avoir le rôle de serveur intermédiaire ou bien le rôle de serveur final. La connexion établie, le client initie une transaction SMTP avec le serveur. Le client et le serveur peuvent ensuite s'échanger des commandes et des réponses conformes au standard SMTP.

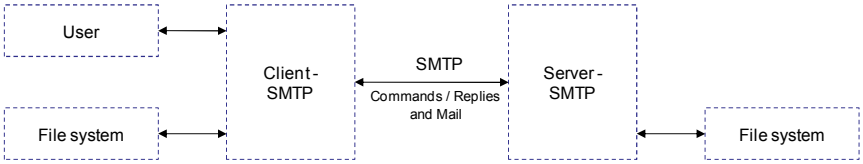


Figure 4 : schéma d'architecture SMTP simplifiée

Pour transmettre un message, le protocole doit dérouler plusieurs phases.

- *Session initiation*. Le client ouvre une connexion avec le serveur et ce dernier répond positivement. Cela correspond à l'ouverture de la session.

- *Client initiation*. Après avoir reçu le message de bienvenue de la part du serveur, le client envoie une commande EHLO indiquant son identité au serveur.
- *Mail transaction*. Cette phase correspond à la transmission de l'enveloppe et du message. Elle est composée de trois étapes.
    - l'envoi de la commande MAIL FROM qui contient l'identification de l'émetteur
    - l'envoi de la commande RCPT TO qui contient l'identification du destinataire. Cette étape est répétée autant de fois qu'il y a de destinataires.
    - l'envoi de la commande DATA qui correspond au début de transmission du message. La transmission du message se termine par une ligne comportant le seul caractère "."
- *Terminating session*. une connexion SMTP se termine lorsque le client envoie la commande QUIT et que le serveur répond positivement.

Ces différentes phases sont illustrées dans le diagramme de séquence de la Figure 5.

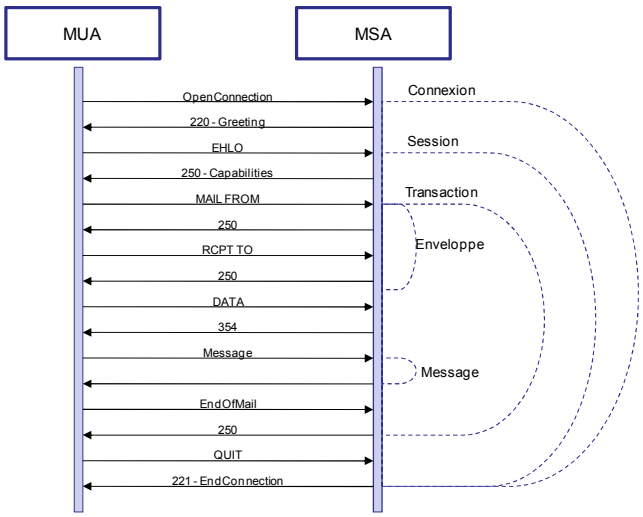


Figure 5 : diagramme de séquence d'une transaction SMTP

Dans une session SMTP, le client envoie des commandes et le serveur transmet en réponse des codes sur trois chiffres pour indiquer le statut. Le premier chiffre indique le statut global. Les deux autres chiffres fournissent des détails. L'IANA maintient un référentiel[18] des codes renvoyés par le serveur SMTP.

- code 2 : la demande a été exécutée sans erreur ;
- code 3 : la demande est en cours d'exécution ;
- code 4 : indique une erreur temporaire ;
- code 5 : la demande n'est pas valide et n'a pas pu être traitée.

Le standard SMTP décrit d'autres commandes optionnelles mais elles ne sont pas abordées dans ce document. Lorsque le serveur SMTP transmet une réponse positive à la fin de la transmission du message, il prend la responsabilité de la remise du message à son destinataire ou

[18] Ce référentiel est disponible à l'adresse http://www.iana.org/assignments/smtp-enhanced-status-codes/smtp-enhanced-status-codes.xhtml

de la transmission à l'émetteur d'un rapport d'échec de transfert du message. Ce point correspond à la notion de fiabilité décrite dans le RFC5321.

Il est important de noter que le protocole SMTP peut être étendu et permettre ainsi aux clients et aux serveurs SMTP de négocier des extensions de fonctionnalités. Ce principe a été mis en œuvre à travers la description d'extensions au protocole SMTP. Cela ne remet toutefois pas en cause le fonctionnement original du protocole. L'extension SMTP peut prendre la forme d'une nouvelle commande ou bien d'un nouveau mot clé ajouté à une commande existante. Les extensions proposées par le serveur sont transmises au client en réponse à la commande EHLO. Ces extensions sont appelées également des capacités. Le serveur peut toutefois ne supporter aucune extension au protocole SMTP. Il existe de nombreuses extensions qui sont en général décrites dans des RFC. Le Tableau 2 fournit une liste non exhaustive d'extensions du standard SMTP :

RFC 1870	SMTP Service Extension for Message Size Declaration
RFC 2505	Anti-Spam Recommendations for SMTP MTAs
RFC 2920	SMTP Service Extension for Command Pipelining
RFC 3030	SMTP Service Extensions for Transmission of Large and Binary MIME Messages
RFC 3461	Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notifications (DSNs)
RFC 3463	Enhanced Mail System Status Codes
RFC 3464	An Extensible Message Format for Delivery Status Notifications
RFC 3834	Recommendations for Automatic Responses to Electronic Mail
RFC 4954	SMTP Service Extension for Authentication

Tableau 2 : Extensions SMTP

Le protocole SMTP permet le transfert d'un message entre deux composants. Les associations possibles sont MUA/MSA, MSA/MTA, MTA/MTA, MTA/MDA, MDA/MS. La transmission exécutée entre le MUA et le MSA est appelée soumission (*submission*). La transmission exécutée entre le MDA et le MS est appelée remise (*delivery*). La transmission exécutée entre le MSA et le MTA ou entre MTA ou entre le MTA et le MDA est appelée transfert (*transfer*).

Lorsque le message est remis dans la boite aux lettres, le destinataire doit le récupérer. Le standard SMTP ne propose pas de solution de récupération de message. Les standards POP3 et IMAP4 ont été définis pour réaliser cela. Le standard POP3 (*Post Office Protocol version* 3), compte tenu de son abandon progressif, ne sera pas traité dans ce rapport.

2.7 *Le standard IMAP*

Le standard IMAP4 (*Internet Message Access Protocol*, Version 4rev1 IMAP4rev1) décrit un protocole qui permet à un client d'accéder et de manipuler les messages présents sur un serveur. Il a été développé en 1986 comme une alternative au protocole POP3 [27]. La version IMAP actuelle est définie dans le RFC3501 [28].

Avec IMAP, un utilisateur peut créer, effacer, renommer des boites aux lettres ou encore effacer, rechercher, sélectionner des messages, voire positionner des drapeaux (flags) caractérisant le statut d'un message (*Seen*, *Draft*, *Recent*…). IMAP propose également des

mécanismes pour manipuler des boites aux lettres localement puis de les synchroniser avec des boites aux lettres distantes. Le protocole IMAP4 est basé sur TCP et un serveur IMAP4 écoute sur le port 143. Enfin, contrairement à POP, IMAP autorise des utilisateurs à être connectés simultanément sur la même boite aux lettres.

Le protocole IMAP4 fonctionne sur le principe de commandes et réponses. Un client établit une connexion avec le serveur IMAP4. Une invitation est renvoyée par le serveur. Ensuite, le client et le serveur interagissent à l'aide de commandes envoyées par le client et des réponses, de données transmises par le serveur ou de réponses envoyées par le serveur. Une fois la connexion établie entre le client et le serveur IMAP, le protocole IMAP4 peut être dans un des quatre états suivants :

- *Not authenticated*. Le client doit s'authentifier avant d'être autorisé à envoyer une commande
- *Authenticated*. Le client est authentifié et doit sélectionner une boite aux lettres avant d'accéder à des messages.
- *Selected*. Le client a sélectionné une boite aux lettres.
- *Logout*. Cet état résulte de l'envoi par le client d'une commande « *LOGOUT* »

Il existe de nombreuses commandes permettant de manipuler les boites aux lettres ou les messages. Elles sont décrites en détail dans le document RFC3501. La commande FETCH permet de récupérer les attributs ou bien les données associées à un message. Un exemple de commande IMAP4 est proposé ci-dessous. Le client demande la récupération du corps MIME de trois messages dont les identifiants sont 2, 3 et 4. Le client demande également le drapeau, ainsi que les champs d'entête *Date* et *From*.

```
C: A654 FETCH 2:4 (FLAGS BODY[HEADER.FIELDS (DATE FROM)])
S: * 2 FETCH ....
S: * 3 FETCH ....
S: * 4 FETCH ....
S: A654 OK FETCH completed
```

Le concept d'enveloppe dans IMAP est différent de celui dans le standard SMTP. L'enveloppe IMAP4 correspond à l'entête du message IMF. Cette enveloppe est créée par le serveur IMAP4. En ce qui concerne le corps du message IMF, il correspond dans IMAP4 à une partie appelée TEXT.

A partir du protocole IMAP4, un message peut être manipulé de la manière suivante :

- IMF correspond à la partie RFC822
- L'entête IMF correspond à la partie RFC822.HEADER
- Le corps IMF correspond à la partie RFC822.TEXT

Il existe de nombreuses extensions au protocole IMAP4. Voici une liste non exhaustive d'extensions :

RFC 2177	IMAP4 IDLE command
RFC 5464	The IMAP METADATA Extension
RFC 5465	The IMAP NOTIFY Extension
RFC 5466	IMAP4 Extension for Named Searches (Filters)
RFC 5550	The Internet Email to Support Diverse Service Environments (Lemonade) Profile
RFC 5593	Internet Message Access Protocol (IMAP) - URL Access Identifier Extension
RFC 6237	IMAP4 Multimailbox SEARCH Extension
RFC 6785	Support for Internet Message Access Protocol (IMAP) Events in Sieve
RFC 6855	IMAP Support for UTF-8

Tableau 3 : Extensions IMAP

Il est important de noter que l'envoi d'un message à partir du protocole SMTP est souvent associé au dépôt d'une copie du message émis dans la boite aux lettres de l'émetteur. Ce dépôt est réalisé à partir du protocole IMAP. Ceci est toutefois configurable et dépendant de la politique locale au MUA.

2.8 *L'acheminement du message*

Les paragraphes suivants décrivent en détail les aspects soumission, transfert et récupération des messages.

2.8.1 **La soumission de message**

Les mécanismes décrits dans le standard SMTP ne permettent pas de différencier les messages émis localement (mécanisme de soumission de message), des messages routés entre serveurs (mécanisme de transfert de message). Le document RFC 5068 [29] souligne que le lien le plus faible dans la chaîne de confiance d'un système de messagerie est celui de la soumission du message. L'application de politiques spécifiques lors de la soumission trouve donc tout son intérêt. Pour répondre à cette limitation, le standard SUBMISSION a été défini.

Grâce à ce nouveau standard, il devient par exemple possible :

- d'appliquer des politiques spécifiques de filtrage afin d'empêcher l'injection de messages non autorisés (spams, virus…),
- d'appliquer des mécanismes d'authentification ne limitant l'accès qu'aux utilisateurs dûment autorisés,
- de détecter des problèmes de configuration de certains clients,
- de mettre en œuvre des extensions de service de soumission,
- …

Le standard SUBMISSION a été défini et décrit dans le RFC6409 [7]. Ce document introduit un nouveau composant appelé MSA (*Mail Submission Agent*). Le MSA agit comme un serveur de soumission pour accepter les messages en provenance du MUA émetteur et agit comme un client SMTP pour relayer le message vers un MTA. Le MSA communique avec le MUA à l'aide des commandes et réponses définies dans le standard SMTP cependant le MSA met en œuvre un port d'écoute dédié (port 587) au processus de soumission. Tout message reçu sur ce port est considéré comme un message de soumission.

La Figure 6 décrit l'architecture mise en œuvre pour permettre une soumission interne et une soumission externe de messages. Le MUA_l est dans le réseau local de la même manière que le MSA et peut réaliser une soumission interne de messages. Le MUA_2 n'est pas localisé dans le réseau local et doit traverser le réseau Internet pour soumettre un message, il réalise dans ce cas une soumission externe.

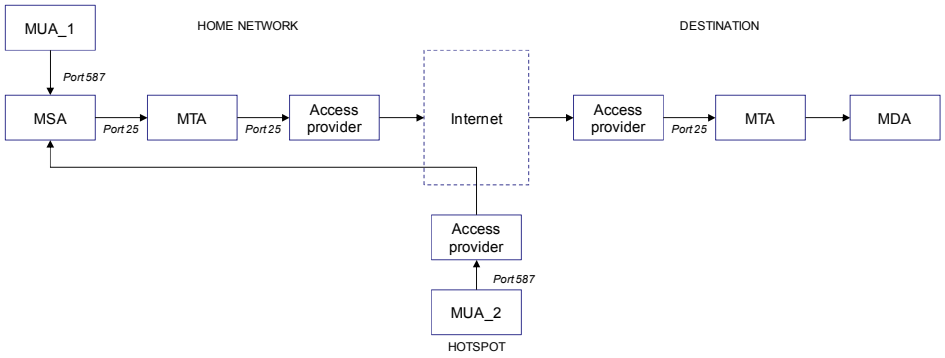


Figure 6 : exemple d'usage du port 587 via Internet

2.8.2 **Le transfert du message**

L'objectif du transfert de message est de transmettre le message du MSA jusqu'à la boite aux lettres du destinataire du message. Ce principe nécessite différentes transmissions entre les composants de l'architecture de messagerie. Ce principe de transmissions successives est appelé "*hop-by-hop*".

Pour transmettre le message, le système de messagerie doit interagir avec le système de nom de domaine, appelé DNS (*Domain Name System*). Le DNS, décrit dans le RF1035 [30], propose un mécanisme de correspondance entre les boites aux lettres et les noms de domaine. Ce mécanisme utilise le nom du domaine présent dans la partie droite (*domain part*) de l'adresse de messagerie du destinataire. Un des avantages de ce mécanisme est de découpler le nom de la destination de l'hôte utilisé pour router le message. Le DNS utilise ensuite le nom de domaine pour localiser la liste des hôtes acceptant le message pour ce domaine. Ceci est réalisé à partir des champs de type MX (Mail eXchanger). Un serveur SMTP désigné dans un enregistrement DNS de type MX n'est pas toujours le composant en charge de la remise finale du message. Il peut avoir le rôle de relai. S'il accepte cette tâche alors après avoir réceptionné le message, il devient un client SMTP et établit une nouvelle session avec le serveur SMTP suivant (spécifié dans le DNS). L'agent qui a le rôle de serveur est en attente sur un port d'écoute dédié, à savoir le port 25. Ce principe d'interaction entre le système de messagerie et le système de nom de domaine est illustré dans la Figure 7.

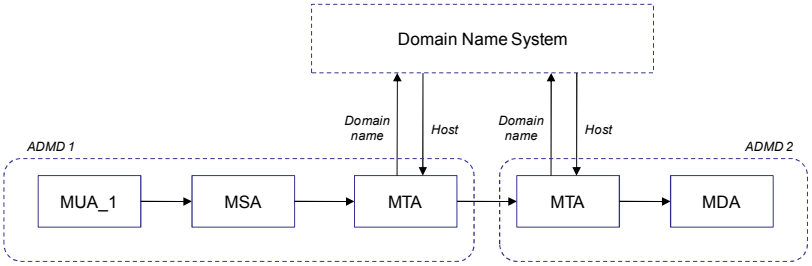


Figure 7 : Interaction entre le système de messagerie et le système de nom de domaine

Lorsqu'un utilisateur envoie un message vers plusieurs destinataires, il n'émet en général qu'un seul message. Toutefois, afin que ce message soit transmis à tous les destinataires, il est dupliqué. En fonction de l'architecture et des destinataires du message, cette duplication peut être réalisée dans différents composants impliqués dans la chaîne de transmission du message. Dans le cas d'une transmission dans un environnement intra-ADMD, le message sera en général dupliqué au niveau du MDA qui est en charge de la remise dans la boite aux lettres des destinataires. Dans le cas d'une transmission dans un environnement inter-ADMD, une duplication sera réalisée entre les MTA de bordure (BMTA ou *Boundary* MTA) des différents domaines. Un envoi peut donc générer plusieurs messages. Le nombre de messages émis est donc fonction du nombre de destinataires.

2.8.3 **La récupération des messages**

Le standard SMTP propose des mécanismes permettant de soumettre (*submission*), de transférer (*transfer*) puis de remettre (*delivery*) le message dans la boite aux lettres du destinataire. Cependant, après ces différentes étapes, le destinataire ne dispose pas directement du message. Il doit accéder, à l'aide de son MUA, à sa boite aux lettres puis récupérer le message. Pour réaliser cette dernière étape, le standard IMAP4 a été défini. Il a été détaillé précédemment.

2.8.4 **Les notifications**

Quand un serveur SMTP accepte un message, il prend la responsabilité de le relayer ou le remettre dans la boite aux lettres du destinataire. Il ne doit pas perdre le message. Le standard SMTP souligne d'ailleurs que son objectif principal est de transférer les messages de manière fiable et efficace.

Cependant, l'acheminement d'un message ne se déroule pas toujours correctement. Pour fournir une certaine fiabilité, le client après avoir rencontré un échec lors de la transmission d'un message doit essayer de le retransmettre. Cette retransmission peut être réitérée cinq fois. Si cette retransmission échoue définitivement, alors le client doit généré une notification et la transmettre à l'émetteur initial du message. Dans un système de messagerie, ce principe de notification est appelé DSN (*Delivery Status Notification*) ou *Bounce*. Tout agent en charge de l'acheminement d'un message doit être capable de transmettre un DSN en cas de problème. Le format d'un DSN est décrit dans le RFC3464 [31].

SMTP propose également de transmettre des notifications à l'émetteur d'un message quand son acheminement se déroule correctement. Pour cela, tous les agents en charge de l'acheminement du message doivent mettre en œuvre l'extension DSN, décrite dans le RFC3461 [32]. Si un

agent n'implémente pas cette extension alors l'émetteur ne pourra pas être informé du succès de l'acheminement complet du message.

Il existe également un mécanisme de notification permettant de transmettre un accusé de remise. Le principe de ce mécanisme est de permettre à un destinataire d'envoyer une notification lorsqu'il télécharge le message ou bien lorsqu'il l'affiche. La notification est envoyée à l'émetteur du message. Ce mécanisme, appelé MDN (*Message Disposition Notification*), est basé sur la présence d'un champ d'entête particulier. Le format du MDN est décrit dans le RFC3798 [33].

2.9 *Le concept d'identité dans les systèmes de messagerie*

Le concept d'identité dans les systèmes de messagerie est important. D. Crocker propose [6] une liste des formes d'identités manipulées dans une architecture de messagerie : l'adresse de messagerie, la boite aux lettres, le nom de domaine et l'identifiant du message.

Une adresse électronique est une chaîne de caractères qui identifie un utilisateur pour lequel un message sera envoyé. La description d'une adresse électronique est fournie dans le chapitre 2.4.

Une boite aux lettres reçoit les messages et correspond à la zone de stockage de ces messages. Une boite aux lettres est spécifiée comme une adresse électronique.

Un nom de domaine (*domain name)* est une référence globale à une ressource Internet (ordinateurs, services, réseau...). Il correspond souvent à une ou plusieurs adresses IP. Il est utilisé pour faire référence à une ADMD.

L'identifiant du message peut prendre deux formes : *message-ID* qui se rapporte au contenu du message et *ENVID* qui se rapporte à son transfert. Chaque message conforme au standard IMF contient un champ *message-ID* qui doit être unique. *ENVID* est un paramètre de la commande MAIL FROM décrite dans le standard SMTP. Il est utilisé dans un but de traçabilité des messages [34] dans un système de messagerie.

2.10 *Bilan*

Le courrier électronique, tel que nous le connaissons aujourd'hui, a vu le jour au début des années 1970. Ce service a subi différentes évolutions pour aboutir à la description des quatre principaux standards actuels, à savoir SMTP, SUBMISSION, IMAP et IMF. Le standard SMTP décrit un mécanisme de transmission de message jusqu'à la boite aux lettres du destinataire. Le standard SUBMISSION, basé sur SMTP, propose des fonctionnalités supplémentaires dédiées à la soumission du message. Le standard IMAP décrit des mécanismes de récupération de message et de gestion de boite aux lettres. Enfin, le standard IMF décrit la structure du message échangé.

En 2009, afin de remédier à des développements pouvant comporter des incohérences et à des mises en œuvre non compatibles, un document permettant de clarifier les architectures de

messagerie a été publié par l'IETF, le RFC 5598 [6]. Ce document présente les grands principes d'architectures de messagerie électronique basées sur les protocoles de l'Internet. Il propose notamment une description globale de l'ensemble des composants, services et utilisateurs regroupés au sein d'une architecture appelée IMA (Internet Mail Architecture). Une architecture IMA propose deux modèles : intra-ADMD (ou intra-domaine) et inter-ADMD (ou inter-domaine). En intra-ADMD, les messages sont transmis entre composants présents dans un même environnement d'administration et utilisant le même nom de domaine. Il y a une seule autorité administrative et tous les composants du système de messagerie sont sous la responsabilité de cette autorité. Dans le cas d'échanges dans un modèle inter-ADMD, les messages sont échangés entre différentes ADMD. L'émetteur et le destinataire du message n'appartiennent pas à la même ADMD et sont sous la responsabilité d'autorités différentes.

Il est important de noter que les standards SMTP, SUBMISSION, IMAP et IMF ne proposent pas de service de sécurité. Cette situation a entrainé l'émergence de nombreuses menaces comme l'hameçonnage (phishing), les messages non sollicités (spam) ou encore l'usurpation d'identité (email spoofing). Afin de répondre à ces différentes menaces, la description et la mise en œuvre de service de sécurisation du courrier électronique sont devenues rapidement nécessaires. Ils sont décrits dans le chapitre suivant.

Chapitre 3

Les concepts de messagerie sécurisée

Les premiers systèmes de messagerie électronique développés, dans le courant des années soixante et soixante-dix, étaient relativement simples et conçus pour une petite communauté. Cette simplicité a permis une appropriation rapide par les utilisateurs et un développement exponentiel de ce service. Cependant, cette simplicité a aussi permis l'émergence de nouvelles menaces qui évoluent rapidement et sont, tous les jours, plus sophistiquées. Au fil des ans, la sécurisation des systèmes de messagerie est donc devenue un objectif critique. Ceci est d'autant plus vrai que la messagerie électronique est largement présente dans les entreprises et les organisations [35] [36].

Ce chapitre décrit en détail les menaces et les techniques d'attaques auxquelles les systèmes de messagerie font face. Il présente également les propriétés de sécurité liées à la messagerie électronique. Enfin, il décrit les solutions de sécurisation et les modèles d'architecture de messagerie sécurisée basés sur les standards actuels.

3.1 ***Menaces, vulnérabilités et techniques d'attaques***

Avant d'analyser en détail les menaces, les vulnérabilités et les techniques d'attaques, il est important de donner une définition à ces trois termes. Nous baserons ces définitions sur les travaux de F. Schneider [37].

A threat is an adversary motivated and capable of exploiting a vulnerability.

An attack is the means of exploiting a vulnerability.

A vulnerability is an error or weakness in design, implementation or operation.

Compte tenu du succès de la messagerie électronique et de son aspect pervasif, les menaces sont nombreuses et peuvent être d'origines diverses [38] [39] [40]. Elles peuvent être le fait d'employés mécontents, de pirates, d'espions, de terroristes, d'agences gouvernementales ou encore de professionnels du crime. Les chiffres et rapports[19][20] publiés par les sociétés spécialisées mentionnent que les attaques restent encore importantes.

En fonction de leurs origines, ces menaces peuvent prendre différentes formes d'attaques [41] :

- messages indésirables (spam),
- hameçonnage (phishing),

[19] https://www.securelist.com/en/analysis/204792334/Spam_in_Q1_2014

[20] http://www.symantec.com/content/en/us/enterprise/other_resources/ b-istr_main_report_v19_21291018.en-us.pdf

- hameçonnage ciblé (spear-phishing),
- usurpation d'identité (email spoofing),
- attaque par homme du milieu (man-in-the-middle)
- écoute clandestine (eavesdropping),
- génération d'un faux message,
- perte d'un message,
- modification des données d'un message,
- rejeu de message,
- canulars (hoax),
- chaînes de messages,
- déni de service (denial of service),
- ...

Les messages indésirables [42] appelés aussi spam (ou "*Unsolicited Bulk Email*", UBE) sont des messages non sollicités qui peuvent prendre différentes formes. Ce peut être des messages commerciaux envoyés par une société. Dans ce cas, nous parlerons de UCE ("*Unsolicited Commercial Email*"). Les techniques d'attaques ont évolué dans le temps. Passant de l'envoi direct de messages puis de l'utilisation de relais ouverts à des techniques plus sophistiquées comme l'utilisation de malwares[21] installés sur des ordinateurs piratés. Le spam peut aussi prendre la forme de campagnes de type "*pump-and-dump*" dont l'objectif est de tenter de faire évoluer le cours en bourse des actions d'une société ou bien d'attaques de type "*scam 419*"[22] ayant pour but de soutirer de l'argent de façon frauduleuse en laissant penser aux futures victimes qu'elles obtiendront un pourcentage d'une somme liée à un héritage, une loterie... Ce qui ne se produit pas.

Les attaques peuvent être massives comme dans le cas des messages indésirables [43]. Elles peuvent être plus sélectives, cas de l'hameçonnage [44] qui peut limiter les destinataires des messages à certains domaines, au niveau d'une extension DNS d'un pays par exemple. L'hameçonnage est une forme de messages non sollicités. Le principe est de tenter de récupérer des informations sensibles comme des numéros de comptes bancaires ou des codes d'accès en donnant l'impression au destinataire que le message provient d'une source sûre et légitime. Les menaces peuvent également être ciblées et associées à des techniques d'ingénierie sociale, cas de l'hameçonnage ciblé. Ce type d'attaque sophistiquée est basé sur l'envoi de messages personnalisés à un faible nombre de destinataires, voire un seul. Ce principe rend cette attaque dangereuse, car le message peut être crédible et très convaincant.

D'autres formes d'attaques existent. L'attaque "*joe job*" est une technique de spam qui permet l'envoi de messages comportant une adresse d'émetteur usurpée. L'objectif initial étant de ternir la réputation de l'émetteur usurpé ou d'entrainer des actions contre lui. L'attaque "*backscatter*" a pour effet d'engorger un système par la réception de notifications (*bounce message*) envoyées automatiquement par des serveurs de courrier à des personnes qui n'étaient pas à l'origine du message initial.

---

[21] Un logiciel malveillant ou maliciel (en anglais, malware) est un programme développé dans le but de nuire à un système informatique, sans le consentement de l'utilisateur infecté.

[22] La dénomination 4-1-9 vient du numéro de l'article du code nigérian sanctionnant ce type de fraude (wikipedia).

L'usurpation d'identité (*email spoofing*) est une attaque classique dans les systèmes de messagerie électronique. Le principe est de modifier l'adresse d'émission du message. Cela est possible grâce à l'utilisation d'une adresse d'émission présente dans l'enveloppe SMTP différente de celle présente dans le champ d'entête du message. Cependant, les serveurs de messagerie moderne fournissent des fonctionnalités de sécurité permettant de vérifier la cohérence de ces adresses.

L'écoute clandestine permet de capturer les informations transmises entre différents protagonistes d'un échange de message. Cela peut être réalisé plus ou moins aisément. Un administrateur pourra facilement copier les messages transitant par le serveur qu'il administre. De la même manière, des transmissions sans fils non sécurisées pourront être écoutées. Un autre risque lié à l'écoute clandestine, est la perte des informations d'authentification auprès d'un service de message, typiquement le login et le mot de passe. Dans ce cas, l'individu malintentionné peut lire les messages reçus et présents dans la boite aux lettres de l'utilisateur mais il peut également envoyer des messages sous l'identité de l'utilisateur abusé.

La modification de messages peut facilement être réalisée par un administrateur qui possède des permissions sur le serveur de messagerie. De la même manière, il peut supprimer certains messages sans que l'émetteur ni le destinataire en soit informés.

La génération de faux messages (on parle de forger un message) fait partie des attaques courantes dans les systèmes de messagerie. Dans les systèmes non sécurisés, il est relativement simple de générer un faux message émanant d'un utilisateur abusé. Il est difficile pour ce dernier de prouver qu'il n'est pas l'émetteur du message, il peut dans ces cas, nier être l'émetteur: on parle de répudiation.

Bien que moins courant, le rejeu de message est un type d'attaque dont l'objectif est de sauvegarder un message puis de le réémettre plus tard en faisant en sorte qu'il apparaisse comme valide.

Il existe également des attaques par déni de service. Ces techniques d'attaques visent à saturer une boite aux lettres par l'envoi massif de messages vers un même utilisateur ou un même serveur. On parle dans ce cas de "*mail-bombing*" et de "*mail-bomb*. La taille limite de sa boite aux lettres atteinte, l'utilisateur visé ne pourra plus recevoir de messages.

Enfin, la messagerie est un vecteur important de transmission de logiciels malveillants (malware) comme les vers, virus... Ils peuvent être directement intégrés dans le message sous la forme de pièces jointes ou bien être transportés sous la forme de lien vers des sites hébergeant des logiciels malveillants.

Les définitions des principales attaques sont proposées dans le RFC4949 [45].

Dans certains cas, la menace ne provient pas d'individu malveillant mais d'une mauvaise utilisation des outils. Prenons le cas, par exemple, de messages comportant des informations relatives à un projet dont les destinataires n'ont pas à avoir la connaissance. L'émetteur peut très bien sélectionner par erreur des homonymes comme destinataires du message. Ce cas se retrouve également dans le domaine militaire où les informations sont liées à un marquant de sensibilité (Confidentiel, Secret...). Tous les utilisateurs ne peuvent pas traiter le même niveau de sensibilité de l'information. Il faut donc se prémunir de ce type d'erreurs non intentionnelles dont les effets peuvent être importants.

Remarque : il faut faire la distinction entre attaques et effets. Par exemple, la modification d'un message aura pour effet une altération du message. L'écoute clandestine aura pour effet une perte de confidentialité ou de non-respect de la vie privée.

3.2 *Propriétés de sécurité*

Afin de garantir la sécurité de l'information, des principes clés ont été définis. Parmi ceux-ci se trouve le triptyque CID[23]; Confidentialité, Intégrité et Disponibilité. Il correspond aux trois propriétés de sécurité initialement définies dans le document ISO[24] 27000 [46] comme nécessaires à la sécurisation de l'information.

"*Information security is the preservation of confidentiality, integrity and availability.*"

Trois autres propriétés, à savoir l'authentification, la non-répudiation et le contrôle d'accès, ont été ajoutées aux propriétés initiales pour former CIA+ [47]. Ces six propriétés s'appliquent aux systèmes de messagerie [48][49][50]. Les définitions, extraites de la norme de l'ITU[25] X.800 [51], de ces propriétés sont les suivantes:

- **confidentialité :** propriété d'une information qui n'est ni disponible, ni divulguée aux personnes, entités ou processus non autorisés.
- **intégrité des données :** propriété assurant que des données n'ont pas été modifiées ou détruites de façon non autorisée.
- **disponibilité :** propriété d'être accessible et utilisable sur demande par une entité autorisée.
- **répudiation :** le fait, pour une des entités impliquées dans la communication, de nier avoir participé aux échanges, totalement ou en partie.
    - **non répudiation avec preuve de l'origine (non repudiation with proof of origin) :** l'émetteur ne peut pas nier être l'auteur du message.
    - **non répudiation avec preuve de la remise (non repudiation with proof of delivery) :** le destinataire ne peut pas nier la réception du message.
- **authentification :** cette propriété est déclinée en deux propriétés : authentification de l'origine des données et authentification de l'entité homologue
    - **authentification de l'origine des données (data origin authentication) :** confirmation que la source des données reçues est telle que déclarée.
    - **authentification de l'entité homologue (peer entity authentication) :** confirmation qu'une entité homologue d'une association est bien l'entité déclarée.
    - **authentification de l'utilisateur (user authentication service) :** confirmation qu'une identité clamée par une entité peut accéder au système.

[23] Ce triptyque est appelé CIA en anglais pour Confidentiality, Integrity et Availability.

[24] L'Organisation internationale de normalisation, ou ISO est un organisme de normalisation international composé de représentants d'organisations nationales de normalisation de 164 pays.

[25] International Telecommunication Union. L'ITU est l'agence des Nations unies pour le développement spécialisé dans les technologies de l'information et de la communication, basée à Genève (Suisse).

- **contrôle d'accès :** protection de ressources d'un système contre des accès non autorisés.

Certains travaux [52] ont défini des propriétés dérivées:

- **Autorisation :** propriété qui permet à une requête d'un utilisateur d'effectuer une action demandée. Cette propriété est invoquée dans le cadre du contrôle d'accès.
- **Imputabilité *(Accountability)* :** propriété qui garantit que les actions d'un utilisateur peuvent être liés précisément à cet utilisateur et qui tient cet utilisateur pour responsable de ses actes.
- **Notarisation :** propriété qui permet d'attester de l'origine, la destination et l'intégrité d'un contenu particulier. De plus, cette propriété peut attester de la date de départ et de réception.

Ces propriétés de sécurité sont présentes dans un ou plusieurs des agents du système (MUA, MSA, MTA, MDA et MS), lors de l'échange du message et directement sur le message.

3.3 *Services de sécurité, solutions de sécurité*

L'objectif des services de sécurité est de mettre en œuvre des mécanismes afin d'apporter des réponses aux différentes menaces. Ces mécanismes permettent d'appliquer les propriétés de sécurité. Il existe des solutions pour répondre aux menaces de l'hameçonnage [53] [54] ou des messages indésirables [55] [56] [57] [58]. Les travaux de cette thèse ne ciblent pas directement la menace des messages non sollicités toutefois les solutions proposées dans ce rapport pourront répondre à certains aspects de ces menaces.

Il n'existe pas d'architecture type permettant de répondre à toutes les menaces. Tenter de définir une architecture type et de l'appliquer aveuglément peut mener à une catastrophe [59]. Il faut donc fournir des solutions de protection pour faire face à la large variété de menaces. La solution découlera d'une analyse qui prendra en compte les menaces et les vulnérabilités. Cela permettra d'en déduire les risques et la criticité de chacun. Par exemple, un réseau non connecté à Internet ne fera pas face aux mêmes menaces qu'un système interconnecté.

Aujourd'hui, les solutions de sécurisation des systèmes de messagerie sont basées en grande partie sur des mécanismes cryptographiques. Ces mécanismes permettent d'assurer et de vérifier les propriétés décrites précédemment. Ils sont décrits en détail dans [52][60]. La mise en œuvre de ces mécanismes cryptographiques impose la distribution des clés publiques, via des serveurs de clés ou par échange direct entre utilisateurs. Ce principe devient rapidement complexe. Les infrastructures à gestion des clés (IGC) ou Public Key Infrastructures (PKI) ont été définies dans un objectif de gestion des clés. Cependant, les services offerts par une PKI sont bien plus larges. Une PKI peut être définie comme un ensemble d'entités, communiquant par des protocoles et offrant des services pour la gestion des clés publiques et des certificats [61].

En fonction des propriétés ciblées, les solutions de sécurité s'appliqueront sur une ou plusieurs des trois catégories listées ci-dessous :

- L'accès au service

- L'acheminement du message
- Le message

3.4 *Sécurisation de l'accès au service*

L'accès au service est important, car il permet à l'utilisateur d'envoyer de nouveaux messages et/ou de gérer ses messages. L'accès au service requiert une authentification et une autorisation. Pour accéder au service, l'utilisateur doit, dans un premier temps, être authentifié. D. Crocker souligne que la confiance dans les systèmes de messagerie commence avec l'authentification [62]. La façon dont les utilisateurs s'authentifient auprès des systèmes est un des sujets les plus anciens et les plus largement étudiés en matière de "sécurité utilisable"[63]. Cela a tout d'abord été réalisé à l'aide d'identifiants et de mots de passe. Ensuite, ces mécanismes ont évolué et la présentation d'une donnée signée numériquement a permis d'apporter des informations nécessaires à l'authentification d'un utilisateur.

L'utilisateur authentifié, s'il dispose des autorisations suffisantes, peut ensuite accéder au système. Un mécanisme basé sur des listes de contrôle d'accès (ACL, *Access Control Lists*) permet ceci. Lorsque les ACL sont construites, elles sont généralement liées spécifiquement à une ressource donnée et contiennent les identifiants de la partie autorisée à accéder à la ressource en question et les actions autorisées à faire sur cette ressource [64].

Les mécanismes d'authentification dans les systèmes de messagerie sont basés sur le standard SASL décrit dans le RFC4422 [65] (*Simple Authentication and Security Layer*). Il fournit une interface entre le protocole SMTP ou IMAP et les mécanismes d'authentification. Grâce à ce principe, n'importe quel protocole peut utiliser les mécanismes d'authentification existants. De plus, de nouveaux mécanismes d'authentification peuvent être ajoutés de façon transparente.

SASL correspond à un cadre qui permet de fournir une couche d'abstraction entre les protocoles et les mécanismes d'authentification. Ce principe est illustré dans le schéma proposé dans la Figure 8, extrait du RFC4422. Par exemple, le protocole SMTP peut s'interfacer avec le mécanisme PLAIN ou bien GSSAPI.

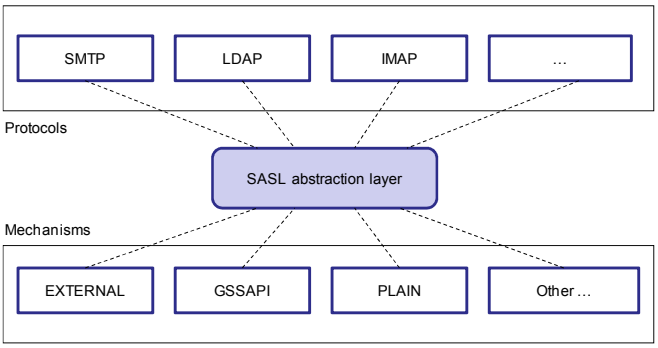


Figure 8 : Principe d'abstraction de la couche SASL

Afin d'utiliser SASL, chaque protocole fournit une méthode pour identifier quel mécanisme sera utilisé, pour définir les informations échangées et pour récupérer le résultat du challenge d'authentification.

Le protocole SMTP dispose d'une extension, appelée SMTP AUTH [66], qui propose une interface avec SASL. L'authentification est requise lorsqu'un utilisateur envoie un message. Cette extension prend la forme d'un nouveau mot clé AUTH présent dans les capacités renvoyées par le serveur. A ce mot clé sont associés les mécanismes d'authentification supportés par le serveur. Le client sélectionne le mécanisme qu'il supporte et débute le challenge d'authentification avec le serveur. Dans le cas où l'authentification aboutit, le serveur transmet une réponse positive sous la forme d'un code. L'exemple ci-dessous décrit une authentification PLAIN négociée entre un client (C) et un serveur (S) qui aboutit à un succès. Dans le cas où l'authentification échoue, le serveur transmet un code d'échec.

```
S: 220-smtp.example.com ESMTP Server
C: EHLO client.example.com
S: 250-smtp.example.com Hello client.example.com
S: 250-AUTH GSSAPI DIGEST-MD5
S: 250-ENHANCEDSTATUSCODES
S: 250 STARTTLS
C: STARTTLS
S: 220 Ready to start TLS
  ... TLS negotiation proceeds, further commands
      protected by TLS layer ...
C: EHLO client.example.com
S: 250-smtp.example.com Hello client.example.com
S: 250 AUTH GSSAPI DIGEST-MD5 PLAIN
C: AUTH PLAIN dGVzdAB0ZXN0ADEyMzQ=
S: 235 2.7.0 Authentication successful
```

Figure 9 : exemple d'une authentification dans une transaction SMTP

Dans certains cas, les mécanismes d'authentification peuvent requérir une confidentialité. Pour répondre à ce besoin, le protocole TLS est utilisé. L'exemple de la Figure 9 décrit une négociation de transaction chiffrée à l'aide du protocole TLS et de la commande STARTTLS. Le standard TLS est présenté en détail dans ce même chapitre.

SMTP-AUTH fournit un mécanisme d'authentification et d'autorisation d'accès au service d'envoi de message. Il ne garantit pas l'authenticité du champ MAIL FROM (présent dans l'enveloppe SMTP) ni celle du champ From (présent l'entête du message). Cette limitation permet à un individu malveillant d'usurper une identité. Cependant, le MSA devrait s'assurer que les informations présentes dans l'enveloppe et dans le message sont conformes et ainsi bloquer les tentatives d'usurpation.

Le protocole IMAP dispose d'une commande AUTHENTICATE qui permet d'indiquer un mécanisme d'authentification basé sur SASL. Si le serveur IMAP supporte le mécanisme d'authentification demandé, il réalise la négociation d'identification et d'authentification. L'identité transmise lors de la négociation d'authentification est interprétée par le serveur comme un nom d'utilisateur ayant des privilèges sur la boite aux lettres associée.

3.5 *Sécurisation du message*

La sécurisation du message, telle que décrit dans ce paragraphe correspond aux mécanismes de sécurisation de bout en bout. L'émetteur applique des mécanismes de sécurisation sur le message et le destinataire les vérifie. Six solutions notables de sécurisation ont vu le jour. Toutes offrent des services d''authentification, d'intégrité, de non-répudiation et de confidentialité du message. Cependant les mécanismes mis en œuvre diffèrent.

En 1987, le groupe PSRG (Privacy and Security Research Group) a développé une technologie nommée PEM (*Privacy Enhanced Mail*). Ces travaux ont été ensuite transférés à l'IETF. Bien que défini comme standard en 1993 par l'IETF [67][68][69][70], il ne fut jamais largement utilisé.

En 1991, P. Zimmerman écrit son propre programme nommé PGP (*Pretty Good Privacy*). Ce logiciel permettait de signer et chiffrer les messages. PGP (depuis renommé en OpenPGP) embarquait également son propre format de certificat numérique et format de stockage. Cependant, le support des certificats X.509 a été ajouté en 2007 [71]. PGP introduit un modèle de confiance décentralisé (contrairement au mode centralisé de la PKI), appelé *Web of trust*. Ce modèle est décrit par A. Abdul-Rahman[72]. Il n'y a pas d'autorité centrale en laquelle tous les utilisateurs font confiance. Les utilisateurs signent chacun d'autres clés et progressivement construisent un domaine de confiance appelé *web of trust*. L'exemple souvent utilisé est le suivant. Alice, une collègue de Carole, signe le certificat de Bob qu'elle sait authentique. Bob transmet son certificat signé à Carole pour communiquer avec elle. Carole qui connait et fait confiance à Alice (considérée dans ce cas comme *introducer*) découvre après vérification qu'Alice est parmi les signataires du certificat de Bob. Ainsi, Carole peut faire confiance à Bob. OpenPGP est une technologie largement utilisée par les particuliers, dans un objectif de respect de la vie privée. En raison de son modèle de confiance, elle n'a pas encore connu de déploiement massif dans l'environnement professionnel.

En 1995, l'IETF a abandonné PEM et a développé le standard MOSS (*MIME Object Security Services*). MOSS [73] était basé sur PEM. MOSS ne fut pas largement utilisé et est maintenant abandonné. Une des principales raisons est l'arrivée du standard S/MIME.

L'armée américaine a défini, à la fin des années 80, un standard de sécurisation des messages : *Message Security Protocol* (MSP). Le développement de MSP s'est déroulé en même temps que celui de PEM. MSP a été conçu initialement pour sécuriser les messages transportés dans un système X.400, mais intégrait un support rudimentaire pour les messages transportés sur des systèmes compatibles Internet. Le NIST a publié la description de MSP [74]. Bien que développé et supporté par certains éditeurs, MSP n'a pas connu de succès commercial.

De son côté, en 1991, la société RSA Data Security[26], spécialisée dans le domaine de la sécurité, a développé PKCS#7 (*Public Key Cryptographic Standard*). PKCS#7 n'avait pas pour objectif d'être dédié à la sécurisation des messages. Son objectif était plus large, la protection de données dans les applications. PKCS#7 a été largement déployé et un consortium a été créé pour porter PKCS#7 à l'IETF. RSA Data Security a céder le contrôle des spécifications PKCS#7 à l'IETF et en 1998, le RFC 2315 a été publié. CMS (*Cryptographic Message Syntax*), présenté dans la suite de ce rapport, est basé sur PKCS#7.

[26] RSA est un sigle formé à partir des noms de ses fondateurs : Ronald Rivest, Adi Shamir et Leonard Adleman. Ce sont les co-inventeurs du crypto système à clé publique du même nom..

Les standards S/MIME et CMS sont décrits en détails dans la suite de ce chapitre. Il faut noter que ces standards sont supportés par la plupart des logiciels de messagerie.

3.5.1 **S/MIME**

Le standard S/MIME propose des mécanismes pour sécuriser le contenu des messages, et plus précisément le contenu MIME qui, pour sa part, ne fournit aucun service de sécurité. S/MIME permet de garantir l'authentification de l'origine, l'intégrité du message et la non-répudiation de l'origine en utilisant les mécanismes de la signature électronique et la confidentialité du message en utilisant les mécanismes de chiffrement. La force de S/MIME représente aussi son défaut. Quand un message est chiffré, il ne peut être déchiffré que par son destinataire. Il ne peut donc pas subir d'analyse antivirale, ni de vérification de contenu pour prévenir la divulgation d'information sensible [75]. La spécification de S/MIME est fournie dans le RFC 5751 [11]. S/MIME spécifie également les conventions d'usage des certificats X.509 par les agents du système de messagerie [76]. Ce standard, développé initialement par la société RSA Data Security, est largement implémenté dans les solutions de messagerie actuelles.

Le standard S/MIME est composé de deux parties. La première décrit le format du message sécurisé et la seconde le traitement lié aux certificats. Pour créer un message sécurisé, S/MIME déroule trois étapes. Tout d'abord, il y a la préparation d'un nouveau type MIME, ensuite il y a une canonisation des différentes parties MIME et enfin l'application d'un mécanisme d'encodage (en général base64[27]). S/MIME décrit le contenu MIME du message sécurisé, mais ne décrit pas la structure du message. Cette description est réalisée par le standard CMS (*Cryptographic Message Syntax*). CMS est une syntaxe pour protéger les messages, indépendante du mode de transport. Dans le cas de S/MIME, l'objet CMS est encapsulé dans MIME. Plusieurs nouveaux types MIME sont définis (cf. tableau ci-dessous).

Type		Extension
application/pkcs7-mime	SignedData, EnvelopedData	.p7m
application/pkcs7-mime	degenerate SignedData certificate management message	.p7c
application/pkcs7-mime	CompressedData	.p7z
application/pkcs7-signature	SignedData	.p7s

Un type de contenu *SignedData* désigne un objet qui contient une signature électronique. Dans des cas particuliers, ce type désigne non pas un objet signature mais un objet comportant des certificats. Le processus de génération d'un type de contenu *SignedData* est illustré dans la Figure 10.

[27] base64 est un codage de l'information utilisant 64 caractères, choisis pour être disponibles sur la majorité des systèmes.

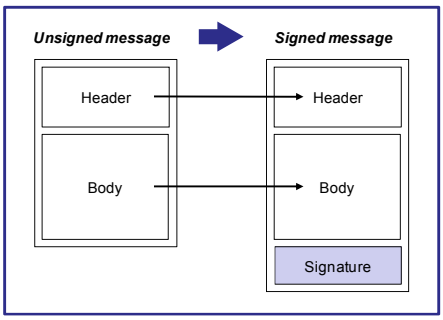


Figure 10 : processus de génération d'un message comportant un type de contenu SignedData

Un type de contenu *EnvelopedData* désigne un objet qui contient des données chiffrées. Le processus de génération d'un type de contenu *EnvelopedData* est illustré dans la Figure 11. Dans certains cas, différents scénarios d'encapsulation de ces types peuvent être envisagés (chiffrement puis signature, signature puis chiffrement...).

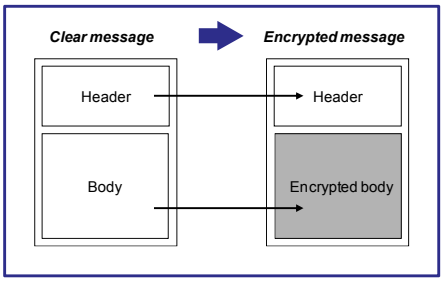


Figure 11 : processus de génération d'un message comportant un type de contenu EnvelopedData

Un type de contenu *CompressedData* désigne un objet sur lequel un mécanisme de compression a été appliqué. Lors de la réception d'un message, l'agent compatible S/MIME procède de la même manière qu'avec MIME. Il analyse le contenu des en-têtes présents dans le message. Dans le cas de la présence de type de contenu S/MIME, il applique les traitements appropriés (vérification de la signature, déchiffrement ou décompression). Il est important de noter que les mécanismes de base de S/MIME ne s'appliquent que sur le contenu du message et non sur l'entête de celui-ci. Afin de protéger la partie entête, il existe une piste proposé dans le standard. Elle consiste en une encapsulation complète du message dans un nouveau corps de message de type message/rfc822. Cependant, cette solution applique les mécanismes S/MIME sans différenciation des champs d'entête, ce qui pose notamment des problèmes dans le cas d'un contenu chiffré.

Il existe des extensions au standard S/MIME et notamment le RFC 2634 [77] qui décrit quatre nouveaux services de sécurité :

- le reçu signé(*signed receipts*),
- les étiquettes de sécurité (*security labels*),
- les listes de distribution sécurisées(*secure mailing lists*),
- la signature de certificats (*signing certificates*).

Grâce à ces nouveaux services, il est possible d'appliquer de nouvelles propriétés. Le reçu signé garantit la non répudiation de la destination. Les étiquettes de sécurité permettent d'appliquer

des mécanismes de contrôle de diffusion en fonction d'autorisation. Ce standard propose également la description d'un scénario d'encapsulation de type de contenu S/MIME appelé triple enveloppe (signature d'un contenu, chiffrement de la signature et du contenu puis signature de l'objet chiffré).

3.5.2 **Cryptographic Message Syntax**

CMS (*Cryptographic Message Syntax*), définie dans le RFC 5652 [78], est une syntaxe d'encapsulation pour la protection de données. Avec CMS, un objet contenant des données peut être encapsulé dans un autre, afin d'appliquer sur ce dernier des mécanismes cryptographique. CMS est transport-agnostique [75]. De la même manière que PKCS#7, sur lequel elle est basée, CMS n'avait pas pour objectif initial d'être dédiée à la sécurisation des messages. Avec CMS, il est par exemple possible de sécuriser un fichier sur un disque dur. CMS supporte les mécanismes de signature électronique et de chiffrement. L'objet CMS est généré en utilisant ASN.1[28] et l'encodage BER[29]. CMS ne vérifie pas si les données à sécuriser sont au format binaire. Le protocole S/MIME applique un encodage et un formatage MIME, afin de permettre le transport d'un objet CMS. CMS spécifie également les conventions d''usage de différents algorithmes cryptographiques [79].

CMS propose la protection d'un contenu qui est identifié par le type de contenu *ContentInfo*. Ce type de contenu peut encapsuler d'autres contenus. Un type de contenu *ContentInfo* est identifié par un oid (*object identifier*). CMS défini six types de contenu qui peuvent être encapsulés :

- *data,*
- *digested-data,*
- *encrypted-data,*
- *authenticated-data,*
- *signed-data,*
- *enveloped-data,*

Le type de contenu *data* est utilisé pour faire référence à des données au format ASCII, comme un fichier texte par exemple.

Le type de contenu *digested-data* consiste en un contenu libre et un condensé du contenu. Ce type de contenu est principalement utilisé pour garantir l'intégrité

Le type de contenu *encrypted-data* consiste en un contenu libre chiffré. Ce type de contenu n'est associé à aucun destinataire et ne dispose pas de clés de chiffrement de contenu (contrairement au type de contenu *enveloped-data*). Ce type de contenu est utilisé pour chiffrer un contenu pour un stockage local avec par exemple une clé de déchiffrement dérivée d'un mot de passe.

---

[28] Abstract Syntax Notation 1 est une syntaxe et une notation qui décrit les règles et les structures pour représenter, encoder, transmettre et décoder des données. ASN.1 est un standard joint ITU-T, ISO et IEC. ASN.1 est défini dans les standards X.680, X.681, X.682 et X.683 de l'ITU-T.

[29] Le codage BER (Basic Encoding Rules) est un des formats d'encodage définie par le standard ASN.1. BER est défini dans le standard X.690 de l'ITU-T.

Le type de contenu *authenticated-data* consiste en un contenu libre, un code d'authentification de message (MAC - *message authentication code*) et des clés d'authentification chiffrées pour un ou plusieurs destinataires. Ce principe permet de protéger en intégrité, tout type de contenu pour un nombre arbitraire de destinataires.

Les deux principaux types de contenu utilisés dans les systèmes de messagerie sécurisée sont *signedData* et *envelopedData*. Dans la suite de ce document, nous décrirons ces deux types en détail.

Le type *signedData* permet de transporter n'importe quel type de contenu et une à plusieurs valeurs de signature. Plusieurs signataires peuvent signer en parallèle le contenu. Cependant, l'application typique d'un type *signedData* est une seule signature numérique d'un signataire qui s'applique sur le contenu. Le principe est basé sur le mécanisme de signature numérique décrit précédemment. Cependant, CMS a quelques particularités. Il autorise l'ajout d'attributs spécifiques (jeton d'horodatage, contre signature...) qui seront associés avec la signature et protégés de la même manière que le message.

Un objet de type *signedData* transporte différentes informations (cf. Figure 12) :

- la version de la syntaxe (*version*),
- une collection d'identifiants d'algorithmes de hachage (*digestAlgorithms*) utilisés par tous les signataires,
- le contenu signé (*encapContentInfo*),
- une collection de certificats (*certificates*) permettant de vérifier le chemin de certification de tous les signataires,
- une collection de listes de révocation (*crls*) permettant de vérifier que tous les certificats présents sont valides,
- une collection *SignerInfos* qui contient les informations pour chaque signataire.

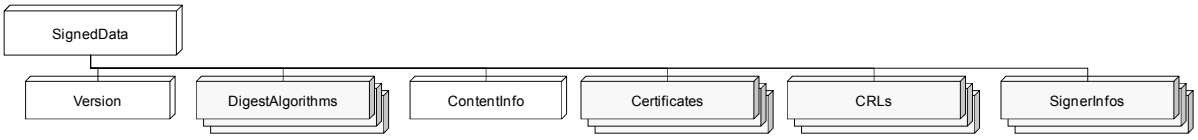


Figure 12 : Description de la structure SignedData de l'objet CMS

Le type de contenu *signedData* est décrit en ASN.1 dans le RFC 5652 [78]. *SignerInfos* est une collection de *SignerInfo*. Il y a autant d'objets *SignerInfo* que de signataires. Un objet de type *SignerInfo* transporte différentes informations (cf. Figure 13) :

- la version de la syntaxe (*version*),
- l'identifiant du certificat du signataire (*signerID*) nécessaire pour la vérification de la signature par le destinataire,
- l'identifiant de l'algorithme de hachage (*digestAlgorithm*) utilisé,
- une collection d'attributs qui seront signés (s*ignedAttrs*),
- l'identifiant de l'algorithme de signature (s*ignatureAlgorithm*) utilisée,
- la signature qui correspond au résultat du processus de génération de la signature et dépend de l'algorithme employé (*signature*),

- une collection d'attributs qui ne seront pas signés (*unsignedAttrs*).

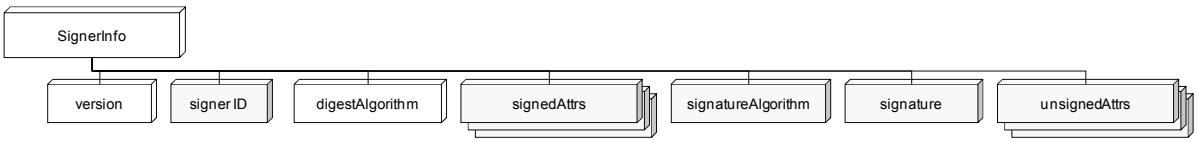


Figure 13 : Description de la structure SignerInfo de l'objet CMS

Les collections *signedAttrs* et *unsignedAttrs* permettent de transporter des attributs avec la signature. La structure d'un message S/MIME comportant un objet CMS de type *SignedData* est illustrée dans la Figure 14.

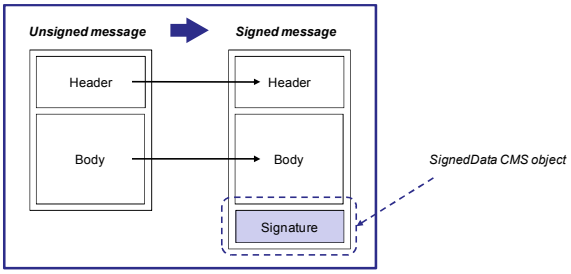


Figure 14 : structure d'un message S/MIME avec un type de contenu SignedData

Le type *envelopedData* consiste en un contenu chiffré et un ensemble de clés de chiffrement, elles même chiffrées en fonction du nombre de destinataires du message. Ce type permet le support du chiffrement d'un contenu d'un émetteur vers plusieurs destinataires. N'importe quel type de contenu peut être chiffré. Un objet de type *envelopedData* comporte différentes informations (cf. Figure 15):

- la version de la syntaxe (*version*),
- des informations concernant l'émetteur (*originatorInfo*) comme des certificats,
- une collection d'informations (*recipientInfos*)pour chaque destinataire,
- le contenu chiffré (*encryptedContentInfo*),
- une collection d'attributs (*unprotectedAttrs*) qui ne seront pas sécurisés mais seront transportés avec l'objet CMS.

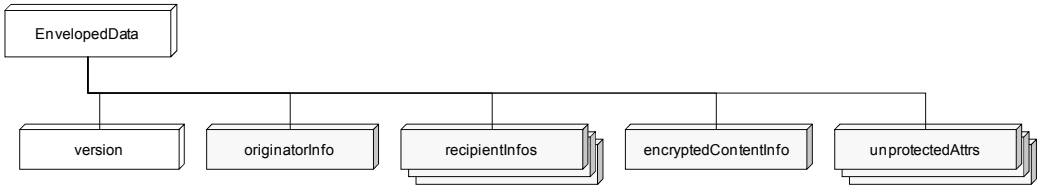


Figure 15 : Description de la structure envelopedData de l'objet CMS

La structure d'un message S/MIME comportant un objet CMS de type *EnvelopedData* est illustrée dans la Figure 16.

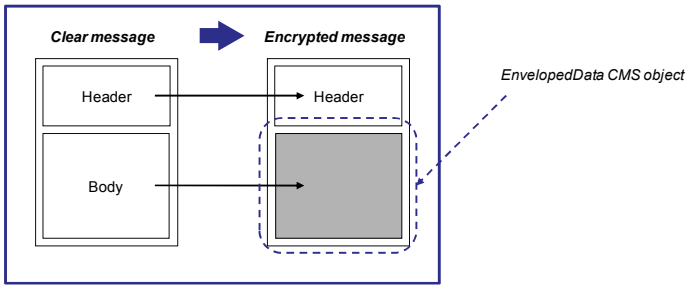


Figure 16 : Structure d'un message S/MIME avec un type de contenu EnvelopedData

3.6 *Sécurisation de l'acheminement du message*

L'application de mécanismes de sécurisation lors de l'acheminement d'un message est un challenge tout aussi complexe que la sécurisation du message. Chaque étape de transmission du message peut être sécurisée et garantir ainsi une sécurisation de l'émetteur jusqu'au destinataire. Cependant, ce principe impose soit une maîtrise de l'architecture ; c'est à dire une capacité à appliquer des politiques dans chaque agent en charge de la transmission du message, soit la mise en œuvre de mécanismes reconnus et acceptés par les différents domaines d'administration (au sens ADMD). Il existe plusieurs solutions qui répondent à des exigences différentes.

SPF (*Sender Policy Framework*) est un standard de vérification du nom de domaine de l'expéditeur d'un courrier électronique. Il est décrit dans le RFC7208 [80]. Il a été développé au sein d'une communauté en charge de la lutte contre le spam. SPF permet à des ADMDs d'explicitement autoriser des agents chargés de la transmission de messages, à utiliser des noms de domaines et à des agents en charge de la réception de pouvoir vérifier de telles autorisations. Les ADMD publient des enregistrements SPF dans les bases DNS spécifiant quels sont les agents autorisés à émettre des messages pour un domaine donné. Les agents de transmission des messages utilisent ces enregistrements pour vérifier l'autorisation d'émission. Le principe proposé ici est dans un premier temps, d'identifier l'émetteur par la présence d'une identité dans les informations de transmission (champs MAIL FROM, HELO...), puis dans un second temps d'authentifier l'émetteur à l'aide du service DNS et enfin dans un dernier temps d'autoriser ou non, la transmission du message. SPF répond à un besoin particulier et ne permet pas, par exemple de vérifier des propriétés d'intégrité ou de confidentialité.

Sender ID (*Sender IDentification Framework*) est un protocole d'authentification créé par le groupe de travail de l'IETF MARID[30]. Sender ID correspond à la fusion des travaux de SPF et de Caller ID. L'objectif de Sender ID est d'authentifier l'expéditeur d'un message, plus précisément le domaine d'expédition. Sender ID est décrit dans le RFC4406 [81]. Sender ID est principalement basé sur SPF avec toutefois quelques particularités afin de répondre à des limitations de SPF. L'identité testée par Sender ID est celle fournie par l'algorithme PRA (*Purported Responsible Address*, décrit dans le RFC4407). Cet algorithme consiste à déterminer l'identité après inspection des champs d'entête suivants du message : Resent-Sender:, Resent-From:, Sender:, From: et à conserver le premier non-vide. Sender ID peut aussi utiliser l'adresse de l'expéditeur indiquée dans la commande MAIL FROM dans la transaction SMTP.

[30] MARID était un groupe de travail en charge en 2004 de proposer des standards pour le sujet "Email authentication". MARID est l'acronyme de MTA Authorization Records In DNS.

DKIM[31] (*DomainKeys Identified Mail*) est un standard d'authentification du nom de domaine de l'expéditeur d'un courrier électronique [82]. DKIM permet à une personne, un rôle ou une organisation qui est propriétaire d'un domaine, de signer un message pour ce domaine. Le message peut être signé par l'auteur du message, mais aussi par un agent de l'ADMD en charge du domaine. DKIM sépare la question de l'identité du signataire du message de l'auteur présumé du message. Le principe de DKIM est basé sur la signature cryptographique du corps du message et d'une partie de ses champs d'entête. L'approche DKIM diffère des autres approches dans le sens où il n'y a pas de dépendance avec une autorité de confiance ou avec une PKI. Une signature DKIM, sous la forme d'un nouveau champ d'entête (DKIM-Signature), permet d'une part d'authentifier l'expéditeur du message, et d'autre part de garantir l'intégrité du message et de certains champs d'entête. Le vérificateur récupère la clé publique associée au nom de domaine de l'émetteur à partir du service DNS. DKIM est décrit dans le RFC6376[83].

Les standard SPF, Sender ID et DKIM ont pour principal objectif de lutter contre le spam et le phishing à partir de mécanismes d'authentification des expéditeurs de message [62]. Ceux-ci ne couvrent qu'une partie des propriétés de sécurité décrites précédemment. Un autre standard, appelé TLS (*Transport Layer Security*), propose une solution de sécurisation du message lors de son acheminement. Il est décrit en détail dans la suite de ce chapitre. Il faut noter que ce standard est supporté par la plupart des logiciels de messagerie.

3.6.1 **Transport Layer Security**

La première solution, proposée par de nombreux fournisseurs et la plus utilisée aujourd'hui, est basée sur le standard TLS (*Transport Layer Security*). TLS[32] est un standard, décrit dans le RFC5246 [10], qui définit un protocole de sécurisation des échanges. TLS fournit un protocole qui permet à une application cliente et une application serveur de communiquer de manière sécurisée, répondant notamment aux menaces d'écoute passive, de génération de faux messages et d'altération de messages. Les protocoles applicatif couramment sécurisés sont HTTP[33] (*Hypertext Transfer Protocol*), TELNET[34] (TErminal NETwork), FTP[35] (*File Transfer Protocol*), SMTP et IMAP. TLS fournit des services d'authentification, d'intégrité et de confidentialité entre deux applications communicantes. TLS est composé de deux protocoles : *TLS Record Protocol* et *TLS Handshake Protocol*. Le protocole *TLS Handshake* est chargé d'authentifier le client et le serveur, de négocier un algorithme de chiffrement et de générer les clés cryptographiques permettant une communication sécurisée. Le protocole *TLS Record* est chargé de protéger les données échangées par l'application. Un des avantages de TLS est d'être indépendant des couches applicatives. Cela permet notamment à la messagerie de mettre en œuvre ce protocole lors de l'acheminement d'un message.

---

[31] Bien qu'associé aux mécanismes de sécurisation appliqués à l'acheminement du message, DKIM applique également une sécurisation sur le message. Cependant son objectif initial, à savoir l'authentification du domaine émetteur, le classe dans cette catégorie.

[32] TLS avait pour prédécesseur Secure Sockets Layer (SSL). Il a été renommé en Transport Layer Security (TLS) par l'IETF à la suite du rachat du brevet de Netscape par l'IETF en 2001.

[33] HTTP est un protocole de communication client-serveur développé pour le World Wide Web Consortium. HTTPS est la variante du HTTP sécurisée par l'usage des protocoles TLS.

[34] Telnet est un protocole réseau utilisé sur tout réseau prenant en charge le protocole TCP/IP. Selon l'IETF, le but du protocole Telnet est de fournir un moyen de communication très généraliste, bidirectionnel et orienté octet..

[35] FTP est un protocole de communication destiné à l'échange informatique de fichiers sur un réseau TCP/IP.

L'authentification proposée correspond à l'authentification de l'entité homologue(*peer entity authentication*). TLS impose l'authentification de l'entité serveur et optionnellement propose l'authentification de l'entité client. L'authentification est basée sur les certificats numériques X.509. L'intégrité des données échangées par l'application est assurée par le protocole *TLS Record* qui utilise un code d'authentification d'une empreinte cryptographique de message avec clé, appelé également HMAC[36] (*heyed-hash message authentication code*). La confidentialité est offerte par le protocole qui *TLS Record*, après génération des clés de chiffrement symétriques par le protocole *TLS Handshake*, chiffre les données échangées par l'application. La description détaillée de TLS est fournie dans le RFC5246 [10].

En raison de son principe de communication sécurisée limitée à l'échange entre une entité client et une entité serveur, TLS n'offre pas un service sécurisé de bout en bout. Lors de l'acheminement d'un message, celui-ci ne sera sécurisé que sur une partie des échanges. Rien ne garantit par exemple, qu'un échange de message soit sécurisé lors de sa transmission entre deux domaines distincts. De plus, le message sera non sécurisé (en clair) à l'intérieur d'un agent de la chaîne d'acheminement du message (MDA, MTA ou MDA). En général, TLS est mis en œuvre lors de la soumission (à l'aide du protocole SUBMISSION) du message par l'émetteur et lors de sa récupération (à l'aide du protocole IMAP) par le destinataire. Cela permet notamment de sécuriser les échanges nécessaires à l'authentification de l'utilisateur. Cependant S. Turner [75] rappelle que lorsque TLS est disponible, il devrait toujours être utilisé.

Dans le cas de la messagerie électronique, il existe deux principes de mise en œuvre de TLS : l'utilisation de ports TCP[37] (*Transmission Control Protocol*) dédiés ou l'utilisation de l'extension STARTTLS. Dans un premier temps, en 1997, l'IANA (*Internet Assigned Numbers Authority*) a enregistré un port dont le numéro était 465 pour l'utilisation de SMTP avec TLS. Le principe proposé était d'utilisé le port 25 pour une communication SMTP non sécurisée et le port 465 pour une communication SMTP sécurisée. En 1998, l'IETF a défini l'extension STARTTLS qui décrit une nouvelle utilisation de SMTP avec TLS. Aujourd'hui, l'utilisation de l'extension STARTTLS, dont la description est fournie dans le RFC3207 [84], est recommandée par l'IETF. D'ailleurs l'IANA, a réassigné le port numéro 465 à un autre type de service, à savoir le multicast audio et vidéo.

Le principe de fonctionnement de l'extension STARTTLS est le suivant. Le client et le serveur initie une nouvelle session. Le client transmet la commande EHLO et le serveur lui renvoie la liste de ses capacités. Parmi celles-ci doit apparaitre le mot clé STARTTLS. Cela signifie que le serveur supporte le protocole TLS. Le client transmet la commande STARTTLS au serveur et ce dernier propose de débuter la négociation TLS. A l'issue du *TLS Handshake*, le client et le serveur peuvent décider de continuer la transaction ou de l'arrêter. Cela peut dépendre de la phase d'authentification. Si la transaction continue, alors le client transmet à nouveau la commande EHLO, ce qui initie une nouvelle transaction sécurisée. Le serveur renvoie la liste de ses capacités qui peut être différente de la liste transmise avant la négociation TLS. Ensuite, le client et le serveur échangent des commandes et des réponses SMTP. Un exemple, extrait du

[36] Un HMAC, de l'anglais keyed-hash message authentication code, est un type de code d'authentification de message, ou MAC en anglais (Message Authentication Code), calculé en utilisant une fonction de hachage cryptographique en combinaison avec une clé secrète. Un code d'authentification de message (MAC, Message Authentication Code) est un code accompagnant des données dans le but d'assurer l'intégrité de ces dernières, en permettant de vérifier qu'elles n'ont subi aucune modification.

[37] TCP est un protocole de transport fiable, en mode connecté, documenté dans la RFC 793 de l'IETF.

RFC3207, est fourni ci-dessous. Les lignes débutant avec le caractère C correspondent aux commandes envoyées par le client et les lignes débutant avec le caractère S correspondent aux réponses transmises par le serveur.

```
S: <waits for connection on TCP port 25>
C: <opens connection>
S: 220 mail.imc.org SMTP service ready
C: EHLO mail.example.com
S: 250-mail.imc.org offers a warm hug of welcome
S: 250-8BITMIME
S: 250-STARTTLS
S: 250 DSN
C: STARTTLS
S: 220 Go ahead
C: <starts TLS negotiation>
C & S: <negotiate a TLS session>
C & S: <check result of negotiation>
C: EHLO mail.example.com
S: 250-mail.imc.org touches your hand gently for a moment
S: 250-8BITMIME
S: 250 DSN
...
```

STARTTLS peut être mis en œuvre avec SMTP et SUBMISSION. Il est d'ailleurs fortement recommandé d'utiliser STARTTLS avec SUBMISSION. Enfin, il faut noter que le serveur peut imposer au client, l'utilisation du protocole STARTTLS. Aujourd'hui, malgré les recommandations, les fournisseurs utilisent encore souvent le port 465.

TLS peut également être utilisé avec le protocole IMAP. Le principe de fonctionnement, décrit dans le RFC2595 [85], est quasiment le même que celui de SMTP. Le client et le serveur initie une nouvelle session. Le client transmet la commande CAPABILITY et le serveur lui renvoie la liste de ses capacités. Parmi celles-ci doit apparaitre le mot clé STARTTLS. Cela signifie que le serveur supporte le protocole TLS. Le client transmet la commande STARTTLS au serveur et ce dernier propose de débuter la négociation TLS. A l'issue du *TLS Handshake*, le client et le serveur peuvent décider de continuer la transaction ou de l'arrêter. Cela peut dépendre de la phase d'authentification. Si la transaction continue, alors le client transmet à nouveau la commande CAPABILITY, ce qui initie une nouvelle transaction sécurisée. Le serveur renvoie la liste de ses capacités qui peut être différente de la liste transmise avant la négociation TLS. Ensuite, le client et le serveur échangent des commandes et des réponses IMAP. Un exemple, extrait du RFC2595, est fourni ci-dessous.

```
C: a001 CAPABILITY
S: * CAPABILITY IMAP4rev1 STARTTLS LOGINDISABLED
S: a001 OK CAPABILITY completed
C: a002 STARTTLS
S: a002 OK Begin TLS negotiation now
<TLS negotiation, further commands are under TLS layer>
C: a003 CAPABILITY
S: * CAPABILITY IMAP4rev1 AUTH=EXTERNAL
S: a003 OK CAPABILITY completed
C: a004 LOGIN joe password
S: a004 OK LOGIN completed
...
```

Le protocole IMAP peut mettre en œuvre TLS à partir d'un port dédié, à savoir le port 993. Cependant, ce principe de fonctionnement n'est pas recommandé. Le principal argument avancé est de ne mettre à disposition qu'un seul port accessible à tous les utilisateurs, que ce soit pour une transaction sécurisée ou non.

3.7 *Bilan*

Afin de répondre aux différentes menaces auxquelles le service de courrier électronique doit faire face, la description et la mise en œuvre de mécanismes de sécurisation sont devenues rapidement nécessaires. Aujourd'hui, les solutions de sécurisation des systèmes de messagerie sont basées en grande partie sur des mécanismes cryptographiques. En fonction des propriétés ciblées, les solutions de sécurité s'appliqueront sur une ou plusieurs des trois couches listées ci-dessous :

- L'accès au service
- L'acheminement du message
- Le message

Afin de limiter l'accès au service de messagerie aux seuls utilisateurs autorisés, le principal standard utilisé aujourd'hui est SASL. Il propose une couche d'abstraction entre les mécanismes d'authentification et les serveurs SMTP et IMAP. En ce qui concerne la sécurisation de l'acheminement du message, le standard TLS a été défini. Il permet de garantir l'authentification des composants en charge de la transmission et le chiffrement des flux d'échanges. Enfin, les entreprises, les organisations et les fournisseurs de services de messagerie utilisent principalement le standard S/MIME associé à CMS pour assurer la sécurisation du message de bout en bout.

Il est important de noter qu'il n'existe pas d'architecture type permettant de répondre à toutes les menaces. Tenter de définir une architecture type et de l'appliquer aveuglément peut mener à une catastrophe. Il faut donc fournir des solutions de protection pour faire face à la large variété de menaces. La solution découlera d'une analyse qui prendra en compte les menaces et les vulnérabilités. Cela permettra d'en déduire les risques et la criticité de chacun. Par exemple, un réseau non connecté à Internet ne fera pas face aux mêmes menaces qu'un système interconnecté. Cette analyse aboutira à la définition et l'application d'une politique de sécurité. Ces aspects sont présentés dans le chapitre suivant.

Chapitre 4

Contrôle de la messagerie électronique par l'application de politiques

Nombreuses sont les organisations utilisatrices de systèmes de messagerie. Certaines de ces organisations ont défini leur propre système répondant à des exigences et des objectifs de sécurité particuliers. Ce principe aboutit en général à la mise en œuvre de systèmes dédiés intégrant des politiques monolithiques. D'autres organisations ont fait le choix de solutions mises à disposition par des fournisseurs de services. Ces solutions proposent en général des services génériques qui ne répondent pas toujours aux besoins des organisations clientes. La mise en œuvre d'un système de messagerie intégrant des services de sécurité et des traitements spécifiques répondant aux exigences d'usages particuliers nécessite la définition et l'application de politiques appropriées. Ce chapitre propose une analyse des principales approches dans le domaine des systèmes de messagerie contrôlés par des politiques.

4.1 ***Contrôle par le destinataire de la remise du message***

L'objectif des travaux proposés par S. Kaushik et al. [86][87] est de remplacer le paradigme actuel de la messagerie par un nouveau. Actuellement, le mécanisme de la messagerie peut être vu comme un droit implicite pour un utilisateur arbitraire d'écrire dans la boite aux lettres de n'importe quel autre utilisateur. La nouvelle approche proposée est de permettre aux destinataires de contrôler la remise de message en boite aux lettres. Ceci est rendu possible grâce à la définition d'une politique de contrôle d'accès basé sur des attributs.

Le problème pointé par S. Kaushik est lié au fait que dans le protocole SMTP, le MTA destinataire (Recipient MTA ou RMTA) agit comme un esclave du MTA émetteur (Sender MTA ou SMTA). L'ajout de mécanismes d'authentification dans SMTP [84] [66] est une amélioration qui ne permet toutefois pas de distinguer un SMTA donné par rapport à un autre SMTA. Kaushik souligne qu'il manque dans les protocoles de messagerie actuels, un cadre pour gérer l'allocation des ressources du destinataire nécessaires à la remise des messages.

Kaushik a défini une architecture de service de messagerie contrôlée par des politiques, appelée PCES (Policy-Controlled Email Services). Celles-ci sont présentées dans la Figure 17.

Les principaux acteurs proposés dans PCES sont :

- Sender
- Sending Email Service Provider (SESP)
- Recipient
- Receiving Email Service Provider (RESP)

L'architecture intègre six types de politiques :

- La SP (Send Policy) est une collection de préférences et d'instructions présentes chez l'émetteur pour répondre aux cas de messages non remis.
- La SPP (SESP Postman Policy) contient les intérêts du SESP et spécifie les exigences de transmission des messages (traitement des virus, priorité de remise...).
- La RPP (RESP Postman Policy) est analogue à la SPP mais dédiée au RESP. Elle contient les intérêts du RESP et spécifie les exigences de remise des messages dans les boites aux lettres.
- La SLAP (Service Level Agreement Policy) décrit comment un RESP décide d'interagir lors de la réception d'un message en provenance d'un SESP.
- La MSP (Message Scheduling Policy) est fournie au SESP par le RESP qui spécifie quels messages devraient être envoyés et quand. Cette politique est optionnelle.
- La MRAP (Message Resource Allocation Policy) est spécifiée par le destinataire. Elle permet de personnaliser les exigences spécifiques d'un destinataire et est utilisée pour déterminer comment les messages seront présentés aux destinataires.

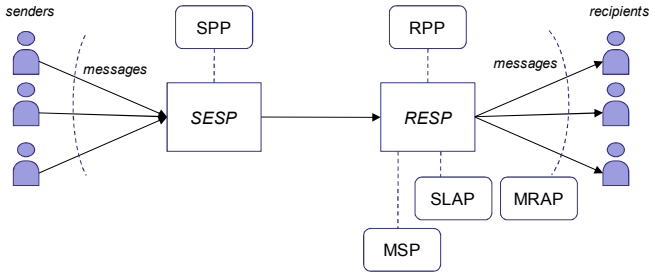


Figure 17 : Composants et politiques du PCES (Policy-Controlled Email Services)

Une extension du standard SMTP est décrite dans PCES. Elle prend la forme d'une nouvelle commande, SHLO, qui remplace la commande actuelle EHLO. SHLO propose les mêmes services que la commande EHLO, cependant elle permet une négociation des politiques en début de transaction SMTP, sous la forme d'un "*handshake*" étendu et la transmission de la MSP.

Les avantages et inconvénients liés à PCES sont les suivants :

- + nouveau modèle qui vise à redonner le contrôle de la remise des messages à leur destinataire
- - la résolution des conflits de politiques ne sont pas abordés dans les travaux
- - Les aspects liés à la mise en œuvre des concepts proposés dans PCES n'ont pas été étudiés
- - le principe de la solution PCES est présenté dans le cadre d'une architecture simplifiée mais son application à une architecture de messagerie de type IMA[6], tout en posant de sérieux problèmes n'est pas abordés dans les travaux (transmission de la politique MSP à travers les différents agents décrits dans IMA).

4.2 *Contrôle de bout en bout via des politiques embarquées associées à une autorité de confiance globale*

D. Chadwick et G. Zhao ont proposé une approche de messagerie sécurisée [88] (*Policy-based Messaging* - PBM) intégrant des mécanismes de relation de confiance distribués. PBM décrit une infrastructure de confiance basée sur des assertions qui permettent de disposer de relations entre le modèle d'application de la politique et les mises en œuvre de ce modèle. Cela permet notamment d'établir des règles et des contraintes dans un environnement ouvert et assure qu'un système distant applique les politiques spécifiées.

Le modèle de confiance PBM est le suivant. L'émetteur du message fait confiance à une autorité qui publie les modèles d'application de politique. Un exemple de modèle d'application de politique proposé est XACML[38] [89]. L'émetteur du message fait également confiance à une ou plusieurs autorités qui certifient que les systèmes destinataires sont capables d'appliquer les politiques ciblées par l'émetteur. Chaque système destinataire certifie, par la diffusion d'un certificat d'attributs X.509 [90], que l'adresse de messagerie du destinataire est bien contrôlée. Enfin, l'émetteur du message est assuré que les messages envoyés à un destinataire qui dispose d'un certificat valide verront leurs politiques appliquées par le système destinataire.

Les principes exposés dans PBM permettent de s'assurer que le message n'est transmis qu'à travers des composants de confiance et donc qu'à travers un chemin de confiance. Pour cela, à chaque étape de l'acheminement du message, le composant émetteur du message s'assure que le composant récepteur du message applique la même politique. Ceci est rendu possible grâce à l'utilisation de différents certificats d'attributs qui sont diffusés à chaque composant du système de messagerie. Dans le cas où un composant ne dispose pas de certificat, alors le message est retourné à l'émetteur avec un avis de non remise.

Dans l'infrastructure PBM, le message est autoporteur de la politique qui contient les règles et contraintes à appliquer. Ce principe d'attachement de la politique à un message est appelé politique attachée ou *sticky policy* [91]. Ensuite cette politique est transportée à travers le système avec le message et doit être appliquée par chaque nœud de traitement présent dans le système.

Les politiques attachées au message sont des politiques qui définissent les usages autorisés et obligations à respecter pour accéder au message. Le client souhaitant accéder à l'information doit négocier avec une autorité de confiance. Dans le cas où la négociation aboutit, l'autorité permet l'accès à l'information en fournissant une clé de déchiffrement du message.

Les avantages et inconvénients à utiliser des politiques attachées sont les suivants:

- + Association directe par un mécanisme d'encapsulation des données du message dans la politique associée. Cela permet de transporter la politique avec le message

[38] XACML (eXtensible Access Control Markup Language) est une spécification qui définit un langage pour le contrôle d'accès, la circulation des règles et l'administration de la politique de sécurité des systèmes d'information. XACML est souvent utilisé pour assurer la fonction d'autorisation dans les architectures SOA.

- + Association forte entre le message et la politique à l'aide de mécanisme cryptographique permettant de vérifier l'intégrité de l'association
- - Les politiques attachées ne peuvent être utilisées que dans le cas ou l'émetteur et le destinataire utilisent une même politique
- - Chaque message est porteur d'une politique avec la possibilité d'une forte redondance. Ceci pourrait occasionner une augmentation du stockage dans les boites aux lettres des utilisateurs.

Le concept majeur de PBM est basé sur la garantie que la politique attachée au message sera appliquée dans le système du destinataire lors de l'accès au message. Cette garantie est apportée par une autorité nommée PIA (*Policy Implementation Authorities*) qui certifie qu'une implémentation particulière est conforme à un modèle d'application de politique. Pour cela, une PIA publie des assertions de sécurité, sous la forme de certificats d'attributs X.509, concernant les capacités du système cible. Il existe trois types de certificats :

- le PEMC (*Policy Enforcement Model Certificate*) qui contient des informations relatives à l'autorité qui publie le certificat et une spécification détaillée du modèle d'application de la politique.
- le PEIC (*Policy Enforcement Implementation Certificate*) qui est diffusé au client et au système cible afin de garantir que le système est conforme au modèle particulier d'application de la politique. Le PEIC est publié par la PIA et il dispose d'une référence universelle du modèle d'application de la politique qui est implémenté par le système cible.
- le EUPC (*End User Policy Certificate*) est publié par le système de messagerie cible. Le champ du porteur du certificat comporte l'adresse de messagerie de l'utilisateur final. Le certificat EUPC lie l'adresse de messagerie de l'utilisateur avec le système qu'il utilise.

Le processus global de PBM inclut la publication des certificats, l'envoi de messages, l'accès aux messages et l'application de politiques. Le processus d'acheminement du message dans PBM est le suivant :

1. L'agent émetteur a besoin d'accéder au certificat EUPC du destinataire et au certificat PEIC du système destinataire. Ceci lui permet de trouver l'ensemble des modèles d'application de politique que le système cible supporte. L'agent émetteur fait ensuite le choix du modèle à utiliser puis compose la politique, basé sur ce qui est défini dans le certificat PEMC. Enfin, l'agent attache la politique au message puis le transmet.
2. Le message est transporté à travers le système de messagerie. Cependant, avant chaque étape de transmission, le composant en charge de l'émission doit vérifier que le composant chargé de la réception dispose d'un certificat PEIC valide et publié par une PIA à laquelle l'émetteur du message fait confiance. Ceci permet de ne pas transmettre le message à un composant ne faisant pas partie d'un chemin de confiance. Si le composant suivant n'est pas de confiance alors le système retourne un avis de non remise à l'émetteur initial.
3. Lorsque le destinataire accède au message, il procède à l'extraction de la politique embarquée. Ceci permet ensuite de définir les règles et contraintes que le système doit appliquer pour le message reçu.

PBM propose un modèle de contrôle d'accès basé sur des politiques embarquées dans les messages. Le système destinataire, en fonction d'accord préalable, applique des mécanismes de

contrôle d'accès sur les messages reçus. Le principe d'autorisation proposé par PBM répond à un besoin particulier des systèmes de messagerie.

4.3 *Contrôle de la qualité de service par des politiques adaptatives*

Le projet AMPol [92] (Adaptative Messaging Policy) est basé sur l'idée que les entités participantes à l'échange d'un message doivent apprendre et adapter les politiques entre elles. L'objectif de AMPol est de supporter des politiques de qualité de service[39] complexes et dynamiques tout en maintenant l'interopérabilité des systèmes. Pour cela, AMPol définit le concept d'adaptation de politique (*policy adaptation*). AMPol définit trois composants fondamentaux:

- le modèle de politique (*policy model*).
- la découverte de politique (*policy discovery*).
- l'extension et l'exécution de politiques (*extension and enforcement*).

Le modèle de politique de AMPol fournit un cadre permettant aux entités impliquées de spécifier un ensemble de règles et de contraintes en fonction des exigences de qualité de service. Il décrit les exigences de qualité de service du comportement dynamique de l'entité sous la forme de politiques.

L'unité de base de AMPol est la règle (*Rule*) qui décrit une exigence d'opération pour un message (*encryption*, *signature*...). Chaque règle est composée de deux parties: action et propriété. L'action spécifie l'opération et propriété comporte les paramètres de l'opération. Les règles sont réunies au sein d'un ensemble de règles (*RuleSet*) et peuvent être combinées grâce à des opérateurs de type OR, AND... Une politique est composé de *RuleSet*. Ces encapsulations sont détaillées dans [92].

Les politiques peuvent être statiques ou dynamiques. La création de politiques dynamiques est réalisée par l'agrégation des politiques présentes dans les différents composants de chaîne de transmission du message. Dans le premier cas, l'entité définit la politique avant que la communication ne se réalise. Dans le second cas, si plusieurs entités sont concernées par la communication, chacune a besoin de connaitre l'ensemble de règles de la politique dynamique. Cela est réalisé à partir du composant de découverte de politiques.

AMPol propose un composant en charge de la découverte de politiques. Grâce à cela, les politiques spécifiées individuellement par les entités peuvent être échangées et permettre une négociation d'un ensemble de politiques acceptables mutuellement. Trois sous-composants sont utilisés pour la découverte de politiques : la publication, la requête et la fusion de politiques.

[39] Il faut prendre la notion de qualité de service comme un ensemble de propriétés non fonctionnelles qui affectent la définition et l'exécution d'un service. Une politique de qualité de service fait référence à un ensemble de contraintes d'un comportement non fonctionnel d'un service.

Après avoir négocié les politiques, les entités peuvent les exécuter. Pour cela, trois sous-composants sont nécessaires : conformité de politique, exécution de politique et extension du système.

Les concepts proposés dans AMPol ont été mis en œuvre dans le projet WSEmail [93] qui propose un système de messagerie basé sur les protocoles de la famille des web services et XML en remplacement des protocoles existants comme SMTP, IMAP et S/MIME.

4.4 *Système de messagerie sécurisée basée sur les rôles*

Les échanges de messagerie dans les entreprises sont souvent réalisés entre personnes. Un message est envoyé à un individu et non à son rôle occupé dans l'organisation. Cependant, l'utilisation de rôle a un vrai intérêt car il n'est pas toujours possible de connaitre les attributions et fonction de chaque individu présent dans une entreprise.

Des services de sécurité sont nécessaires quand les informations présentes dans les messages sont confidentielles. Ces services de sécurité doivent pouvoir être mis en œuvre à la fois pour des personnes mais également pour des rôles dûment autorisés.

Deux solutions permettant de mettre en œuvre une messagerie supportant les rôles existent. L'utilisation d'une boite aux lettres dédiée et l'utilisation de listes de distribution. La première solution impose le partage de mot de passe par les personnes occupant un rôle, ce qui devient très rapidement problématique (nombre d'individus occupants le rôle, changement de rôle, retrait du rôle, changement de mot de passe...). La seconde solution pose également un certain nombre de problèmes. Les messages sont copiés dans les boites aux lettres des occupants du rôle. Après le retrait du rôle, l'individu accède toujours aux messages précédents. De plus, les nouveaux occupants n'accèdent pas aux messages transmis avant l'accès au rôle. Enfin, il existe des problèmes qui ne sont résolus par aucune des deux solutions. Comme un message envoyé par un rôle qui doit être signé par un rôle ou encore un message chiffré à destination d'un rôle qui doit être déchiffré par tous les individus occupant le rôle.

D. Chadwick et al. [94] et G. Zhao [95] ont défini les concepts d'un système de messagerie sécurisée basé sur des rôles organisationnels (*Secure Role Based Messaging*, RBM). L'objectif proposé est de répondre aux limitations précédemment présentées. Le rôle est décrit comme un modèle abstrait qui définit un ensemble de fonctions et de tâches qui peuvent être associées à un ensemble de personnes d'une organisation. L'attribution de rôle doit être dynamique et instantanée, notamment quand un individu est retiré d'un rôle. Le modèle de RBM [96] est basé sur les modèles d'infrastructure de clés publiques (PKI) et d'infrastructure de gestion des privilèges [97] (PMI).

Les travaux proposés sont basés sur l'utilisation de certificats d'attributs X.509 [90] et d'une infrastructure PERMIS [97]. Un nouveau composant d'architecture, le *Role Gatekeeper* (RG), est conçu pour assurer un service de messagerie basée sur les rôles. Le principe proposé est de positionner le RG entre les composants standards que sont les MUA et les MTA et MS et d'intercepter tous les messages entre ces composants. Ce concept d'architecture est illustré dans

la Figure 18. L'objectif du principe d'architecture proposé est de limiter les impacts sur les systèmes de messagerie et les standards existants.

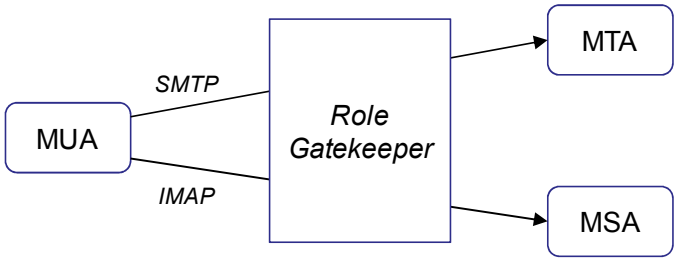


Figure 18 : Architecture RBM

Le composant RG assure des services d'authentification, d'autorisation, de chiffrement et de déchiffrement. Le service d'authentification vérifie l'identité des utilisateurs. Le service de chiffrement et de déchiffrement met en œuvre des mécanismes de signature, de chiffrement et de déchiffrement des messages en accédant à des clés privées associées aux différents rôles. Le service d'autorisation est interfacé avec une architecture PERMIS afin de vérifier si les utilisateurs sont autorisés à réaliser des actions pour un rôle donné. Les autorisations sont décrites au sein de politiques à l'aide du langage XML.

Les utilisateurs disposent de clés privées qui sont stockées localement alors que les clés privées du rôle sont présentes dans le RG et ne sont pas accessible en dehors du RG. Ce dernier accède aux clés privées du rôle à la place de l'occupant d'un rôle pour signer, chiffrer ou déchiffrer les messages à l'aide du standard S/MIME. Les clés publiques des utilisateurs et des rôles sont issues de certificats X.509 [98] et publiées dans des annuaires LDAP [99] (*Lightweight Directory Access Protocol*).

L'assignation d'un rôle est réalisée par une autorité, appelée autorité d'attributs, qui créée et signe un certificat d'attributs. Un certificat d'attributs est une structure de données qui contient des attributs pour une entité donnée. Dans le cas présent, le certificat contient le rôle occupé par un individu. La vérification de l'occupation d'un rôle est possible en vérifiant que le certificat d'attribut a bien été généré par l'autorité.

Trois scénarios sont possibles:

- un utilisateur envoi un message sécurisé à un rôle
- un rôle envoi un message sécurisé à un utilisateur
- un rôle envoi un message sécurisé à un rôle

4.5 *Solutions alternatives*

MailRecall, produit initialement par la société Authentica, fournit des plug-in destinés aux MUA dans le but d'appliquer des politiques permettant de contrôler les accès aux messages, même après leurs remises. MailRecall propose également des mécanismes permettant de déterminer une durée de vie des messages suivie d'une expiration basée sur des politiques de sécurité. Ces

politiques peuvent être configurées localement ou bien centralisées. MailRecall peut être utilisé avec des utilisateurs externes à l'organisation utilisatrice. Dans ce cas, l'utilisateur externe peut accéder au message après s'être enregistré auprès du site et avoir téléchargé un plug-in via son navigateur.

Basé sur le même principe que MailRecall, Omniva Policy Manager est un plug-in destinés aux MUA. Il permet de définir des durées de vie des messages et de contrôler les accès aux messages. L'émetteur peut également limiter ou interdire les transferts de messages, les impressions, les copies de messages. Pour cela, l'utilisateur définit des politiques de sécurité à partir de son client de messagerie.

RMS (Right Management System) est un service développé par la société Microsoft dont l'objectif est la protection des données. Il couvre les contrôles d'accès à tous types de documents comme les documents bureautiques, les messages, les pages web... Cette technologie permet de chiffrer les informations stockées dans ces formats de documents, et grâce à des politiques intégrées dans les documents, ne permettre l'accès au contenu déchiffré que par des personnes autorisées et sous certaines conditions. Les opérations spécifiques comme l'impression, la copie, la modification, la transmission et la suppression peuvent être autorisés ou interdits par les auteurs de contenu.

4.6 *Bilan*

PBM propose un modèle de contrôle d'accès basé sur des politiques embarquées dans les messages. Le système destinataire, en fonction d'accord préalable, applique des mécanismes de contrôle d'accès sur les messages reçus. Le principe d'autorisation proposé par PBM répond à un besoin particulier des systèmes de messagerie.

PCES propose un nouveau paradigme de messagerie électronique. L'approche proposée est de permettre aux destinataires de contrôler la remise de message en boite aux lettres. Ceci est rendu possible grâce à la définition d'une politique de contrôle d'accès basé sur des attributs. Bien que séduisant, ce nouveau modèle n'a pas fait l'objet d'une étude liée à la mise en œuvre.

AMpol propose un modèle de politiques statiques et dynamiques qui permet une négociation entre entités du système et une fusion des politiques présentes dans chacune des entités. Cette solution répond à un besoin particulier. De plus, AMPol est basé sur une architecture orientée services (SOA). Ce principe ne permet pas de faire évoluer un système de messagerie basé sur un socle réunissant les standards SMTP, SUBMISSION et IMAP.

Grâce aux concepts proposés dans RBM, il est possible d'étendre un système de messagerie compatible avec les standards SMTP et IMAP, et d'y ajouter un service de messagerie sécurisée basée sur les rôles. RBM permet d'assigner des rôles à partir de certificats d'attributs X.509 et de contrôler ceux-ci via un système d'autorisation nommé PERMIS. Bien que séduisant, RBM répond à un besoin limité lié à des échanges entre rôles.

Ces différentes solutions ne répondent pas au besoin précédemment exprimé, à savoir la définition d'un système de messagerie intégrant des services de sécurité et des traitements spécifiques répondant aux exigences d'usages particuliers. Nous proposons de définir un

modèle qui permet la définition et l'application de politiques appropriées en fonction du contexte et d'usages particuliers. Ceci est présenté dans le chapitre consacré aux contributions.

Chapitre 5

L'intention de communication dans la messagerie électronique

Le concept de communication interpersonnelle correspond à un échange d'informations entre deux ou plusieurs personnes. **L'acte de communication interpersonnelle** est constitué d'un message envoyé et d'un message reçu. Il est considéré comme réussi quand l'émetteur et les destinataires comprennent le message. D. Sperber et D. Wilson soulignent qu'une communication est réussie non pas lorsque les auditeurs reconnaissent le sens linguiste de l'énoncé, mais lorsqu'ils infèrent le "vouloir dire" du locuteur [100]. P. Hartley ajoute qu'une communication interpersonnelle est partiellement ou totalement intentionnelle [3]. On parle alors d'**intention de communication**.

La messagerie électronique est un outil qui permet de réaliser des communications interpersonnelles de manière asynchrone entre un émetteur et un ou plusieurs destinataires. Il serait intéressant de disposer d'un système de messagerie offrant des capacités de détermination de l'intention de communication de l'émetteur. Cela permettrait l'application de politiques adaptées aux différents usages que les utilisateurs font de la messagerie.

Dans ce chapitre, nous présentons les principaux travaux pouvant contribuer à la détermination des intentions de communication dans les systèmes de messagerie électronique.

5.1 *Le concept d'intention de communication*

Dans le domaine de la psychologie du langage [101], Searle souligne que la communication ne se limite pas à une reconnaissance des mots d'une langue donnée mais impose une compréhension de ce qui est transmis. Il décrit l'intention de communication comme l'intention de «faire reconnaître au lecteur ce que je veux dire, de me comprendre» [102]. L'utilisateur souhaitant communiquer va être motivé par une **intention de communication** qui se situe avant la réalisation de l'acte communiquant cette intention [103]. Dans le cas de la messagerie électronique, l'envoi d'un message sera précédé d'une phase d'initialisation d'un acte de communication de l'utilisateur. Il est possible d'associer cette initialisation d'acte de communication avec la création d'un message électronique. L'utilisateur doit disposer d'un service de messagerie capable de fournir différents types de service en fonction de son intention de communication. Nous définissons cette intention sous l'acronyme *uic* (User Intention of Communication).

S. Frydrychowicz, dans une étude [104] définissant le rôle de l'intention dans le processus de communication interpersonnelle, souligne qu'un participant efficace est capable de transmettre son intention. L'application de ce concept dans un système de messagerie contribuerait à l'apport de nouveaux services.

J. Girao [105] propose le concept de *Intentity* qui agrège l'intention ou la tâche à réaliser avec l'entité finale de la communication représentée par une identité virtuelle. Un des avantages de ce concept est l'association de sécurité de bout en bout plutôt que par couche ou application. Il souligne que le concept d'*Intentity* met l'intention de l'utilisateur au centre.

L'intention de communication peut être vue comme un objet abstrait qui englobe les différents attributs du futur message. Parmi ceux-ci se retrouve des informations courantes comme l'auteur, le (ou les) destinataire(s), l'objet du message, l'heure de création, le caractère d'importance du message ou encore le contenu du message. Ces informations sont présentes dans le standard IMF qui décrit le format des messages. Cependant, l'intention de communication peut aussi comporter des informations spécifiques à l'environnement professionnel. Le service de messagerie doit offrir des mécanismes qui permettent la prise en compte de l'intention de communication.

5.2 *Les usages de la messagerie électronique*

Avant de décrire le concept d'intention de communication, il est nécessaire d'aborder le principe d'usage dans les systèmes de messagerie. L'usage a fait l'objet de différents travaux de recherche. Des premiers travaux [106] ont eu pour objectifs d'étudier l'usage de la messagerie dans une entreprise par rapport aux services déjà existants comme le courrier papier, le téléphone, le fax... ceci afin de proposer des recommandations d'utilisation. Le projet de recherche DLM[40] 3.0 propose de son côté une distinction des usages en trois classes [107] : *corporate email*, *marketing email* et *personal communications*. Le premier cas correspond aux échanges professionnels, le second aux messages ayant un objectif de démarchage commercial et le dernier cas répond au besoin d'échanges privés. D'autres travaux se sont focalisés sur les expériences des utilisateurs lors de l'utilisation de la messagerie dans un environnement professionnel afin de comprendre comment l'usage s'inscrit dans le travail quotidien [108] [109]. L'analyse des échanges dans un environnement professionnel peut, par exemple, induire une expérience positive ou négative [110] et entrainer une disparité des rapports que les individus peuvent entretenir avec cet outil [111]. Le service de messagerie tend, de plus, à être utilisé comme un outil de gestion d'informations personnelles[112][113] et plus seulement comme un outil de messagerie.

Gomez-Skarmeta et al. [114] soulignent que deux personnes dans une même entreprise peuvent avoir différents modes de communication quand elles échangent entre elles. Elles peuvent échanger des informations professionnelles mais aussi des données privées, ce qui nécessite des exigences de sécurité et de protection de la vie privée différentes.

Dans ce rapport, nous définissons l'usage comme le type d'échange que l'utilisateur initie en fonction de son intention de communication. On distingue deux grandes familles d'usage : les usages à caractère privé et les usages à caractère professionnel. Ce dernier englobe de nombreux cas dans le sens où les échanges professionnels peuvent revêtir de nombreuses formes différentes. La messagerie peut, par exemple, permettre l'échange de messages entre autorités administratives de l'état [115]. Dans ce cas, cet usage imposera le respect de certaines exigences décrites dans le référentiel général de sécurité appelé RGS [5]. Les systèmes de messagerie électronique sont également présents dans des environnements spécifiques comme le domaine

[40] Demain Le Mail

militaire [116]. Ils sont désignés sous l'acronyme MMHS[41]. Le domaine militaire impose des contraintes et des exigences fortes décrites dans le standard STANAG[42] 4406 [117] publié par l'OTAN. A l'échelle du réseau Internet, nous pouvons en déduire que les usages de correspondance peuvent être variés et qu'il est impossible d'en réaliser une liste exhaustive. Comme nous venons de le voir ci-dessus, l'usage est important dans la détermination de l'intention de communication.

5.3 *Systèmes de messagerie basés sur la théorie des actes de langage*

Les intentions de communication pourraient être obtenues à partir de l'analyse d'actes de langages dont la théorie a été proposée par J. L. Austin en 1975 [118] puis développée par J. Searle [4] [119]. Selon J. Searle, pour comprendre le langage, il faut comprendre les intentions du locuteur[43]. Puisque le langage est un comportement intentionnel, il devrait être traité comme une forme d'action. L'acte de langage est défini comme un moyen mis en œuvre par un locuteur pour agir sur son environnement par ses mots: il cherche à informer, inciter, demander, convaincre, promettre, etc... son ou ses interlocuteurs par ce moyen. L'acte de langage permet « d'encapsuler » dans un énoncé ce que l'agent dialoguant à l'intention de communiquer [103].

Selon Austin, il y a trois différents types d'actions (ou actes):

- les actes locutionnaires sont liés à la production et à la réalisation des énoncés.
- les actes illocutionnaires sont liés à l'objet ou l'intention d'un énoncé (par exemple, le fait de demander quelque chose, le fait de répondre, en promettant, en refusant, confirmant...).
- les actes perlocutionnaires sont liés aux conséquences de l'auditeur reconnaissant la locution et le but illocutoire de l'énoncé.

En 1987, T. Winograd et F. Flores [120] introduisent le concept de LAP (Language/Action Perspective) qui prend le langage comme la dimension principale de l'activité coopérative humaine. Ce concept stipule que le langage n'est pas seulement utilisé pour l'échange d'informations mais qu'il permet d'effectuer des actions, (comme des commandes, des demandes...) et que ce principe doit être pris en compte dans la conception de systèmes d'information pour disposer d'outils permettant de coordonner des activités communes.

En 1988, T. Winograd[121] propose une première taxonomie des intentions orientées actions, dans les échanges de messages où l'intention appropriée pourrait être manuellement sélectionnée par l'émetteur de chaque message.

---

[41] Military Message Handling System. MMHS est une spécification militaire définie dans l'ACP 123 [185] ainsi que dans le STANAG 4406 [111]. Cette spécification décrit des extensions à la norme X.400 (1992) [154].

[42] Un STANAG est un document de normalisation de l'OTAN

[43] En linguistique et en communication, le locuteur est la personne qui produit des paroles formant un message oral adressé à un destinataire ou interlocuteur.

En 2004, W. Cohen décrit de son côté, une autre taxonomie basée sur la théorie des actes de langage[122]. Ces travaux proposent un cadre pour modéliser les intentions présentes dans le message, ceci grâce à une observation passive des messages et une classification automatique. Un acte [123] est décrit comme un couple verbe-nom qui peut correspondre à une demande, un rappel.... Un message peut comporter de multiples actes.

V. Carvalho [124] propose une taxonomie des intentions en termes d'actes de langage applicables à la communication via la messagerie. Il introduit la notion *Email acts*. Les *Email acts* sont des paires qui associent un nom et un verbe (nom/verbe) et permettent d'exprimer des intentions typiques aux échanges de messages. On peut retrouver, par exemple, une demande d'information, une proposition de réunion, l'engagement pour la réalisation d'une tâche... Le système proposé par V. Carvalho permet une détection automatique des intentions dans les messages et mène à une meilleure gestion et priorisation des messages.

Un message peut contenir de multiples intentions ou actes, et chacun est décrit par une paire nom/verbe. Une taxonomie a pu être définie à partir d'un corpus de 15.000 messages issus de l'université de Carnegie Mellon. La taxonomie, extraite de [124], est la suivante.

Request	A request asks (or orders) the recipient to perform some activity. A question is also considered a request (for delivery of information).
Propose	A propose message proposes a joint activity, i.e., asks the recipient to perform some activity and commits the sender as well, provided the recipient agrees to the request. A typical example is an email suggesting a joint meeting.
Commit	A commit message commits the sender to some future course of action, or confirms the senders' intent to comply with some previously described course of action.
Deliver	A deliver message delivers something, e.g., some information, a PowerPoint presentation, the URL of a website, the answer to a question, a message sent "FYI", or an opinion.
Amend	An amend message amends an earlier proposal. Like a proposal, the message involves both a commitment and a request. However, while a proposal is associated with a new task, an amendment is a suggested modification of an already-proposed task.
Refuse	A refuse message rejects a meeting/action/task or declines an invitation/ proposal.
Greet	A greet message thank someone, congratulate, apologize, greet, or welcomes the recipient(s).
Remind	A reminder message reminds recipients of coming deadline(s) or threats to keep commitment.

Comment classer automatiquement les actes présents dans les messages ? Les messages subissent d'abord un traitement de "nettoyage" qui entraine le retrait des pièces jointes ou des informations présentes dans les champs d'entête qui ne sont pas liées au champ objet. Ce traitement est réalisé manuellement. Ensuite, il y a une extraction des données présentes sous forme de token suivi par l'application d'algorithmes spécifiques d'analyse des données (*VPerceptron, AdaBoost, SVM, DTree*). L'application de ces algorithmes était dans un premier temps limitée au contenu du message. Ensuite, l'ajout d'une analyse des flux (threads) de messages a permis d'affiner le mécanisme de classification. Par exemple, l'analyse d'un message en réponse à un message permettait d'affiner la classification du message initial.

Le concept des actes de langage appliqué à la messagerie électronique pose toutefois le problème des langues utilisées. L'analyse des actes de langage doit être capable de traiter plusieurs types de langues, les spécificités et les particularismes de chacune.

5.4 *Semantic email*

L. Mc Dowell et al. ont introduit la notion de *semantic email* [125][126][127] et décrit comment les concepts de web sémantique [128] pouvaient être appliqués au service de messagerie. Ils ont notamment orienté leurs travaux sur le traitement automatisé de tâches habituellement réalisées manuellement, comme l'organisation d'une réunion ou bien la préparation d'un séminaire. L'objectif est d'automatiser la compilation des réponses positives et négatives afin d'obtenir la liste exhaustive des participants. Le modèle proposé dans ces travaux permet de capturer le contenu sémantique d'un message et les traitements qu'il définit. Pour cela, le concept de *semantic email processes* [129] (SEP) a été introduit. Il permet de questionner des utilisateurs, de collecter leurs réponses et de s'assurer que les résultats satisfont à l'ensemble des objectifs de départ.

Un SEP est constitué de trois composants principaux :

- *l'originator*. Il est l'initiateur d'un SEP. Lors de la création du message, l'*originator* est guidé à l'aide d'une interface graphique dédiée. Cette interface permet également de suivre l'avancement d'un SEP.
- le *manager*. L'*originator* invoque un nouveau SEP en envoyant un message au *manager*. Ce dernier transmet ensuite des messages à tous les participants, il traite les réponses et effectue les modifications en fonction des objectifs de l'*originator*. Le manager stocke toutes les données relatives au SEP, en RDF[44] (*Resource Description Framework*).
- les *participants*. Ils répondent aux messages reçus qui concernent un SEP.

Pour la création d'un SEP, l'*originator* peut créer un modèle de SEP à l'aide d'un editor dédié ou disposer de modèles écrits en OWL[45] (*Web Ontology Language*). Lors de l'exécution d'un SEP, le message reçu par un participant comprend une requête qui mappe les réponses textuelles en RDF. La Figure 19 décrit les interactions entre les composants impliqués dans un SEP.

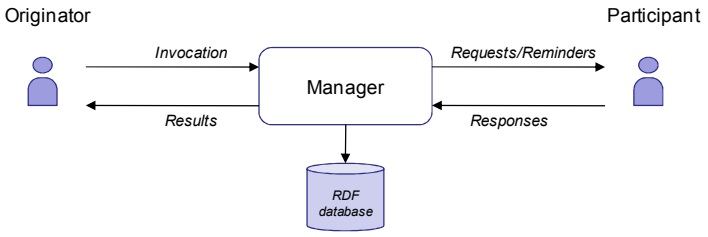


Figure 19 : Invocation et exécution d'un SEP

D'autres travaux de messagerie sémantique ont été réalisés dans le cadre du projet SEMANTA [130][131]. L'approche de SEMANTA permet de combiner la théorie des actes de langage avec les mécanismes d'analyses sémantiques de messages électroniques.

---

[44] RDF est un ensemble de recommandations du W3C (World Wide Web Consortium). RDF est un langage qui permet d'exprimer des modèles de données sous forme d'objets et de leurs relations. http://www.w3.org/RDF

[45] OWL est un langage de représentation des connaissances construit sur le modèle de données de RDF.

L'architecture de SEMANTA est constituée de quatre composants :

- service d'analyse de texte qui permet une annotation semi-automatique du contenu des messages.
- le poste de travail sémantique social qui fournit un stockage RDF dans lequel SEMANTA peut lire et écrire
- une interface graphique simpliste
- un service de messagerie sémantique

Le principe proposé par SEMANTA est le suivant. Lorsque l'utilisateur a terminé la rédaction d'un message, SEMANTA exécute une annotation automatique et une recherche des actions présentes dans le message créé. Un assistant sous la forme d'un plugin propose ces annotations à l'utilisateur qui les valide, les modifie ou les annule. Les actions annotées peuvent correspondre à une demande d'informations, à la planification d'une réunion ou bien encore à l'assignation d'une tâche... Ces actions sont ensuite intégrées dans la partie entête du message qui est envoyé vers le destinataire. Lors de la réception du message, SEMANTA détecte les actions liées au message et exécute les traitements associés à l'action présente. Un message peut par exemple être retourné automatiquement en réponse à une action.

Le projet SEMANTA a démontré qu'il était possible d'extraire automatiquement des informations sous forme d'une représentation sémantique de chaque message et d'exécuter des traitements en fonction d'actions particulières.

5.5 *Les messages semi-structurés*

Des travaux menés par T. Malone [132][133][134] ont permis de définir des messages semi-structurés comme des types identifiables de messages qui contiennent chacun un ensemble connu de champs. L'usage peut prendre la forme de champs additionnels dans les messages. Le concept de messages semi-structurés permet :

- des traitements automatiques des messages dans le système de messagerie,
- une faible contrainte dans la saisie des messages par les utilisateurs. Certains champs restant libres,
- l'apport d'une aide à la saisie pour les champs additionnels grâce à des modèles dédiés,
- l'application de règles par le destinataire,
- une meilleure maîtrise du patrimoine informationnel de l'organisation par l'ajout de métadonnées.

La compréhension du contenu des messages a été très rapidement un défi face auquel les chercheurs ont tenté de trouver des réponses. En 1987, afin de résoudre le problème de partage d'information au sein des organisations, l'idée de structurer l'information grâce à l'introduction de modèles semi-structurés est proposée dans le cadre du projet *Lens*[133]. Le principe est de fournir à l'émetteur du message et à son destinataire, les mêmes modèles de partage d'informations afin de faciliter le filtrage et la catégorisation des messages. Le prototype *Lens* introduit des modèles permettant de distinguer différents types de messages comme l'annonce

d'une réunion ou les « messages à but » (message routé vers un destinataire qui est en charge d'un sujet). Il y a cinq points clés qui forment la base du système d'information *Lens*:

- un ensemble de types de messages semi-structurés peuvent former la base d'un système "intelligent" de partage d'information,
- un ensemble de règles peut être utilisé pour spécifier un traitement automatique de messages,
- l'utilisation de types de message semi-structurés et la définition des règles pour leur traitement automatique peuvent être grandement simplifiées par un ensemble cohérent d'éditeurs orientés affichage pour la composition des messages, la construction des règles et la définition de nouveaux modèles de messages,
- La définition et l'utilisation de messages semi-structurées et des règles de traitement sont simplifiées si les types de messages sont basés sur des principes d'héritage entre eux,
- La mise en place initiale et l'évolution ultérieure d'un tel système de communication peuvent être facilitées si les processus sont réalisés en une série de petits changements.

L. Mc Dowell et al. [127] soulignent que les champs d'entête présents dans les messages sont susceptibles d'être porteurs de données sémantiques. Ce principe d'ajout de données dans un format semi-structuré permet une classification des messages offrant par exemple une protection contre la fuite d'informations [135], une rétention des messages garantissant une manipulation des messages pour une période donnée ainsi qu'une gestion et un contrôle étendus des messages [136].

5.6 *Le contexte et l'intention de communication*

Avec l'usage, le contexte est une donnée importante du concept de correspondance électronique. Usage et contexte sont étroitement liés. La connaissance du contexte permettrait de déterminer les politiques à appliquer pour chaque échange. Semblant dans un premier temps relativement simple, le contexte est souvent bien plus complexe et riche qu'il ne laisse penser [137]. La notion de contexte peut être appréhendée de différentes manières. Le projet *reMail* [138][139], par exemple, associe le contexte au flux conversationnel et permet de faire une relation entre différents messages ayant le même sujet ou d'autres attributs en commun (émetteurs, objet...). Un client de messagerie adapté a été développé pour offrir un affichage des messages en fonction de ces relations [140]. Le contexte peut également prendre d'autres formes comme le statut de présence d'une personne afin de la contacter directement à l'aide d'un service de messagerie instantanée [141].

Les travaux de K. Dey [142] ont défini le contexte comme l'ensemble des informations de situation implicite d'une entité. La définition du contexte sur laquelle nous baserons notre modèle est extraite de ces travaux :

« Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves. »

Cependant, l'utilisation des informations issues du contexte n'est possible que dans le cas où les composants d'un système de messagerie sont capables d'être « sensibles » au contexte ou encore

context-aware, concept apparu dans les travaux de B. Schilit [143][144]. Une définition de ce concept est proposée par G. Abowd et al. [145].

« A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task. »

D. Morse et K. Dey [146] rappellent que les applications de type *context-aware* permettent des exécutions automatiques de services. Ce principe trouve tout son intérêt pour un système de messagerie qui pourrait offrir des capacités de traitements des messages en fonction du contexte. En se basant sur ces mêmes travaux, nous pouvons identifier certains types d'informations de contexte comme appartenant au contexte primaire, à savoir la localisation, l'identité, l'activité et le temps (temporel). Ces types sont appelés primaires, car ils répondent aux questions de base *who*, *what*, *when* et *where*. D'autres types, dits secondaires, peuvent être déduits de ces types primaires. La date de naissance, l'adresse de messagerie ou encore le numéro de téléphone font partie de ces types secondaires.

Des travaux proposent un mode de classification du contexte. Prekop et Burnett [147] font une distinction entre une dimension externe et une dimension interne. De son côté, Hofer [148] distingue le contexte physique du contexte logique. La dimension externe, qui est à rapprocher de l'aspect physique, peut être mesurée par des capteurs matériels. Il est possible de récupérer les informations de localisation, de température, de sons, de lumières... La dimension interne, correspondant à l'aspect logique, est quant à elle, spécifiée par l'utilisateur. Le but, les tâches, le contexte professionnel sont quelques exemples de la dimension logique du contexte.

5.7 ***Bilan***

Nous avons présenté dans ce chapitre le concept de communication interpersonnelle appliqué au système de messagerie électronique. Un acte de communication est précédé d'une intention qui doit être transportée dans le message, échangé entre l'émetteur et le (ou les) destinataire(s). Ainsi, les protagonistes de l'échange disposent d'un même niveau d'interprétation de l'usage qui est fait du système de messagerie. Un message de type professionnel émis par un utilisateur doit être interprété comme un message de même nature par le destinataire. Différentes pistes permettent de déterminer l'usage induit par l'intention de communication. Dans la suite de ces travaux, nous proposons de nous appuyer sur les capacités sémantiques offertes par les champs d'entête présents dans les messages conformes au standard IMF ainsi que sur les concepts de messages semi-structurés proposant une description sémantique du message global.

Le choix du standard IMF est motivé pour plusieurs raisons. A ce jour, il est l'unique standard de description de message électronique. Il est totalement adapté pour être interfacé avec le standard SMTP. Enfin, il permet l'ajout de métadonnées et offre des capacités de définition de messages semi-structurés.

Partie 2

Contributions

Chapitre 6

Le concept de correspondance électronique

La communication interpersonnelle d'un locuteur vers un interlocuteur fournit le cadre général pour aborder notre problématique de l'évolution de la messagerie électronique. Nous identifions l'intention de communication du locuteur, comme un paramètre clé de toute communication interpersonnelle. L'analyse de ce concept nous conduit à introduire la notion de correspondance électronique dont une description fine est proposée au moyen de son cycle de vie. Ce dernier constitue l'outil adéquat montrant comment l'intention de communication peut être dérivée en des politiques applicables dans les différents évènements le constituant.

6.1 *Introduction*

De façon schématique, une communication interpersonnelle mono-message et unidirectionnelle permet à un locuteur de transmettre un message à un interlocuteur via un media (cf. Figure 20).

Figure 20 : Schéma simplifié de la communication interpersonnelle

L'intention de communication du locuteur est le préambule nécessaire qui permet à ce dernier d'amorcer la création du message. Lorsque la communication est réalisée, elle peut être qualifiée à trois niveaux différents:

- la bonne réception du message qui dépend des propriétés du media,
- la bonne compréhension du message qui est liée au caractéristique du message,
- la bonne compréhension par l'interlocuteur de l'intention de communication du locuteur qui est une abstraction de l'étape précédente.

L'instanciation de ces concepts dans un système de messagerie permet de définir les notions suivantes:

- le locuteur est l'auteur du message que nous pourrons parfois désigner par le terme d'émetteur,
- l'interlocuteur correspond au destinataire du message que nous pourrons parfois désigner par le terme de récepteur,
- le media prend la forme du système de messagerie,
- le message correspond au message électronique.

Les systèmes de messagerie électronique existants permettent de qualifier les deux premiers niveaux. En effet,

- la bonne réception du message correspond à son acheminement jusqu'au destinataire final. Nous parlerons alors de communication réalisée. La communication est non réalisée si le message ne parvient pas jusqu'au destinataire final.
- La bonne compréhension du message nécessite son affichage. Nous parlerons de communication conforme. La communication est non conforme si le destinataire ne peut pas visualiser le message.

La détermination du caractère réussi d'une communication est une fonctionnalité absente des systèmes de messagerie existants. L'évaluation de l'intention de communication pourrait être abordée en utilisant certaines approches présentées dans l'état de l'art que nous pouvons qualifier d'a posteriori dans la mesure où elles consistent à analyser le message après sa création et/ou sans opération particulière de l'émetteur, pour en extraire différentes caractéristiques. On peut citer dans cette catégorie, les systèmes de messagerie sémantique basés sur l'analyse des actes de langage.

L'approche que nous proposons dans cette thèse fait l'hypothèse que l'émetteur annonce a priori son intention de communication. Les messages semi-structurés nous offrent un cadre syntaxique intéressant mais insuffisant. La saisie assistée et facilitée du message, le niveau d'interopérabilité par la structuration des messages représentent quelques avantages liés à l'utilisation de messages semi-structurés. Cependant, ces concepts ne proposent pas de prise en compte de politiques en fonction de l'intention de communication de l'auteur d'un message.

Nous introduisons le concept de correspondance électronique qui offre une solution originale au problème de l'établissement de la réussite d'une communication électronique. Nous proposons de qualifier le troisième niveau dans un système de messagerie comme la bonne compréhension par le destinataire de l'intention de communication de l'émetteur. Dans nos travaux, nous associons l'intention de communication à un type d'échange explicitement choisi par l'émetteur. Une communication sera réussie si le destinataire détermine l'intention de l'émetteur (sous la forme d'un type de message) et que les politiques appliquées lors de la transmission et l'affichage du message sont satisfaites. Dans le cas contraire, la communication est non réussie. La classification en trois niveaux que nous proposons est illustrée dans la Figure 21. Ces concepts représentent la base de nos travaux.

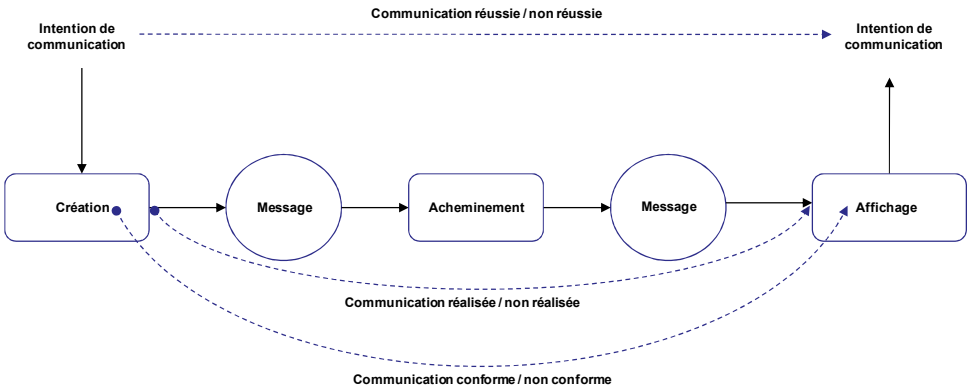


Figure 21 : Qualification des niveaux de communication dans un système de messagerie

6.2 *Détermination de l'intention de communication*

Le cycle de vie du message est initié par une action de création de message. Cette action est précédée d'une intention de communication de la part de l'émetteur. Il est donc nécessaire d'identifier cette intention avant de décrire en détail le modèle de correspondance. Pour cela, nous nous appuyons sur l'exemple proposé dans la Figure 22.

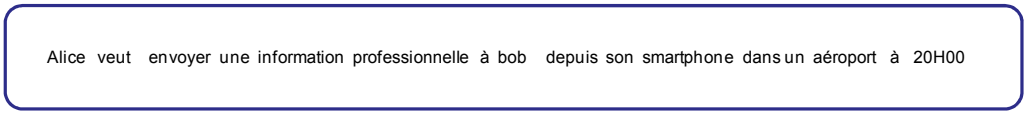


Figure 22 : Exemple d'une intention de communication

A partir de cet exemple, il est possible de distinguer les informations liées à l'intention de communication, des données liées au contexte. En se basant sur les travaux de D. Morse et K. Dey [146] présentés dans l'état de l'art, cette distinction peut être réalisée à partir des réponses aux questions suivantes : *who*, *what*, *when, how, where* et *when*. Il existe une particularité pour le *who* dans le sens où il répond à deux questions: *who* (de qui) et *for whom* (pour qui).

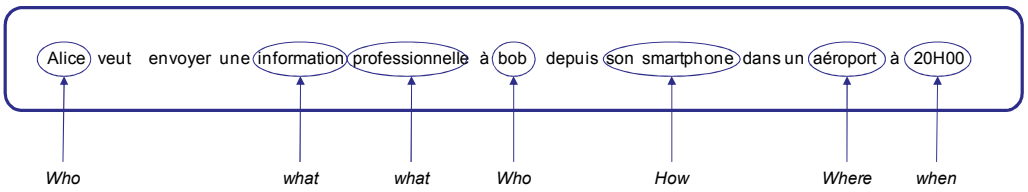


Figure 23 : exemple de décomposition de l'intention de communication

Nous avons vu que les réponses aux questions *how*, *where* et *when* correspondent au contexte externe (ou dimension externe) car mesurable par des capteurs. Les réponses aux questions *who* et *what* correspondent au contexte interne (ou dimension interne).

En reprenant l'exemple, nous pouvons répartir les informations de la manière suivante :

- Contexte interne
    - Alice
    - Bob
    - Information
    - Professionnelle
- Contexte externe
    - Aéroport
    - Smartphone
    - 20H00

Cette classification permet de séparer les informations qui sont liées à l'intention de communication, des informations qui ne le sont pas. Dans la suite des travaux, nous distinguerons les informations du contexte interne en les réunissant au sein d'un objet nommé UIC correspondant à l'intention de communication. Les informations du contexte externe seront relatives au contexte. L'intention de communication est liée à un domaine administratif (au sens ADMD) alors que le contexte est indépendant du domaine.

6.3 *Le message*

Dans les systèmes de messagerie actuels, le message correspond à un objet qui est transmis d'un émetteur vers un ou plusieurs destinataires. Ce message est issu de l'intention de communication de l'émetteur. Nous supposons que l'intention de communication est un objet abstrait qui englobe les différents attributs du futur message. Parmi ceux-ci se retrouve des informations courantes comme le (ou les) destinataire(s), l'objet du message et le contenu du message. Cette intention se transforme en message conforme au standard IMF, lorsque l'auteur renseigne le formulaire proposé par son client de messagerie et que le processus de génération est exécuté.

Cependant, l'intention de communication peut aussi comporter des informations spécifiques qui ne sont pas présentent dans IMF. Par exemple, le niveau de sensibilité du message, le nom du projet, les règles de diffusion du message... Comme nous l'avons vu précédemment, L'ensemble de toutes les informations issues de l'intention de communication sont réunies au sein de l'objet *UIC*. L'intention de communication va permettre de déterminer les formulaires à utiliser pour intégrer dans le message, les attributs spécifiques à celle-ci. Le processus de création d'un nouveau message à partir de formulaire spécifiques revient à transposer l'intention de communication de l'utilisateur dans ce message.

Dans notre modèle, chaque évènement a en entrée ou en sortie, un objet que nous appelons *eo*. Cet objet est constitué d'une partie enveloppe et d'une partie message, cette dernière étant elle-même composé d'une partie entête (*header*) et d'une partie corps (*body*). La structure d'objet *eo* est décrite dans la Figure 24.

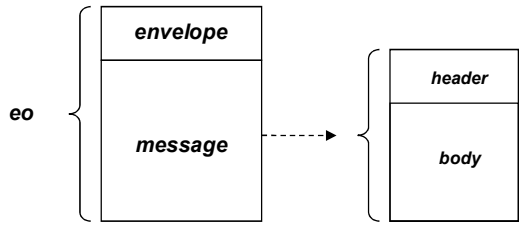


Figure 24 : Format d'un objet message de type eo

L'objet *eo* est porteur d'une information qui fait référence à la correspondance électronique. La présence de cette référence permet une distinction entre les différentes correspondances électroniques garantissant ainsi l'application de traitements et de politiques particuliers. Cette référence est étroitement liée à l'intention de communication de l'émetteur.

6.4 *Modèle de la correspondance électronique*

L'approche habituelle dans le domaine de la messagerie électronique est de disposer d'une politique unique et de l'appliquer dans le système, quel que soit l'usage qui en est fait. Afin de remédier à cette limitation, nous proposons un modèle permettant d'appliquer des politiques appropriées en fonction de l'usage et du contexte. Pour cela, nous introduisons le concept de correspondance électronique. Le concept de correspondance que nous proposons dans cette thèse peut être défini comme un modèle abstrait de la communication interpersonnelle appliqué au service de messagerie. Avant de décrire en détail le cycle de vie d'une correspondance électronique, nous allons poser quelques définitions.

Pour atteindre son destinataire, le message est transporté à travers le système de messagerie et manipulé par différents composants du système. Nous parlerons de cycle de vie du message pour désigner ce principe de transport du message. Le cycle de vie du message (CV) correspond à une séquence ordonnée d'évènements. Le cycle de vie débute par l'évènement de création. Un cycle de vie est complet lorsqu'il se termine avec l'évènement affichage (affichage du message par le destinataire). Dans le cas contraire, il est dit incomplet. L'enchainement des évènements du cycle de vie est décrit dans le chapitre 6.4.2.2.

Nous définissons une **correspondance comme un modèle abstrait qui permet de définir et d'appliquer des politiques appropriées sur le message et lors de sa transmission pour chaque usage et en fonction du contexte courant de l'émetteur**.

Une correspondance est constituée d'une ou plusieurs séquences de cycle de vie. Il n'y aura qu'une seule séquence dans le cas où elle ne comporte qu'un destinataire. Elle sera constitué d'autant de séquences qu'il y a de destinataires. Une **correspondance élémentaire** formalise une séquence du cycle de vie d'un message.

La forme la plus simple de correspondance élémentaire est dite **neutre** dans le cas où les politiques appliquées dans les agents se limitent à celles décrites dans les standards SMTP, IMF et IMAP.

Nous parlerons de correspondance **typée** lorsque des politiques autres que celles décrites dans les standards SMTP, IMF et IMAP sont définies et appliquées.

Une correspondance élémentaire est dite **réalisée** si le message est acheminé jusqu'au destinataire final. Dans le cas contraire, la correspondance élémentaire est dite **non réalisée**.

Une correspondance élémentaire est dite **conforme** si elle est réalisée et que l'utilisateur visualise correctement le message. Ceci inclut la capacité d'interprétation du formatage du message par le MUA. Dans le cas contraire, la correspondance élémentaire est dite **non conforme**.

Une correspondance est dite complètement réalisée quand toutes ses correspondances élémentaires sont réalisées. Nous parlerons de correspondances élémentaires partiellement réalisée lorsqu'au moins une de ses correspondances élémentaires est réalisée. Dans le cas où aucune des correspondances élémentaires n'est réalisée alors la correspondance est dite non réalisée.

Une correspondance est dite complètement conforme quand toutes ses correspondances élémentaires sont conformes. Nous parlerons de correspondances élémentaires partiellement conforme lorsqu'au moins une de ses correspondances élémentaires est conforme. Dans le cas où aucune des correspondances élémentaires n'est conforme alors la correspondance est dite non conforme.

Un correspondance est constituée d'un message initial, appelé aussi message racine. Il peut faire l'objet d'une réponse ou bien d'un transfert vers un autre destinataire. Dans ce cas, les correspondances ayant pour origine une même correspondance sont réunies au sein d'une **correspondance globale**. La correspondance globale (CG) est composée de l'ensemble des correspondances ayant pour origine le même message racine. Bien que ne faisant pas partie des travaux de cette thèse, il est cependant important de noter que les notifications peuvent également faire partie d'une CG. On retrouvera dans ce cas, les types de notification suivants; *Message Disposition Notification*[46] [33], *Delivery Status Notification*[47] [32] [31] et *Signed Receipt* [77].

6.4.1 Cycle de vie d'une correspondance électronique

L'analyse de l'architecture proposée dans le document IMA [6] permet la description du cycle de vie d'un message. D'un point de vue macroscopique, le cycle de vie (CV) correspond à une séquence ordonnée d'évènements. La Figure 25 décrit cet enchainement d'évènements. Un évènement reçoit des données en entrée sur lesquelles il applique des traitements et transmet les données en sortie. Les évènements macroscopiques du cycle de vie sont :

- Création
- Acheminement
- Affichage

[46] Ce type de notification est également appelé accusé de lecture

[47] Ce type de notification est également appelé avis de remise

6.4.2 **Cycle de vie détaillé d'une correspondance électronique**

Afin de définir de manière détaillée le cycle de vie d'un message, nous proposons deux descriptions du cycle de vie : une décomposition hiérarchique et une décomposition globale. Pour des raisons de lisibilité, les cycles de vie proposés comportent uniquement les évènements. Une description détaillée du cycle de vie avec les objets échangés est proposée en annexe I. Afin d'être le proche des standards, les noms des évènements sont en anglais.

6.4.2.1 *Décomposition hiérarchique*

Le cycle de vie macroscopique, illustré dans la Figure 25, est composé de trois évènements : *Creation, Transmission* et *Display*.

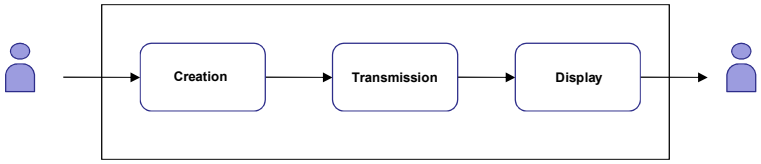


Figure 25 : Cycle de vie macroscopique d'un message

Nous faisons la supposition suivante : l'intention de communication de l'utilisateur est injectée dans le système de messagerie à partir de l'évènement *Creation* puis extrait du système à partir de l'évènement *Display*. *Creation* réalise la génération du message en fonction des données saisies, des propriétés sélectionnées par l'auteur du message et de la politique locale. *Display* permet la visualisation du message et la vérification des propriétés présentes dans le message ainsi que le contrôle de la politique locale. Ces deux évènements correspondent à un échange dit de bout en bout. Le message est acheminé à travers le système via l'évènement *Transmission*. Le niveau d'abstraction de ce modèle permet d'envisager différents mécanismes de transmission comme le système de messagerie bien sur mais aussi le transfert de fichier ou encore l'utilisation d'un support amovible (clé USB par exemple).

L'évènement *Transmission* est une composition des évènements *Routing* et *Retrieving* (cf. Figure 26). *Routing* permet la transmission du message jusqu'à la boite aux lettres de l'utilisateur. *Retrieving* correspond au téléchargement du message à partir de la boite aux lettres de l'utilisateur.

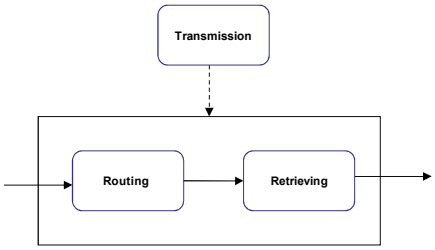


Figure 26 : Décomposition de l'évènement Transmission

La Figure 27 décrit l'évènement *Routing*, composé de trois évènements : *Submission* , *Relay* et *Delivery*. L'évènement *submission* est associé à un transfert de responsabilité entre l'environnement de l'émetteur du message et le système de messagerie. Il permet la soumission du message vers le système. L'évènement *delivery* correspond, pour sa part, à un transfert de

responsabilité entre système de messagerie et l'environnement du destinataire du message. Il correspond à la remise du message en boite aux lettres. Ces transferts de responsabilité sont décrits dans les travaux IMA [6].

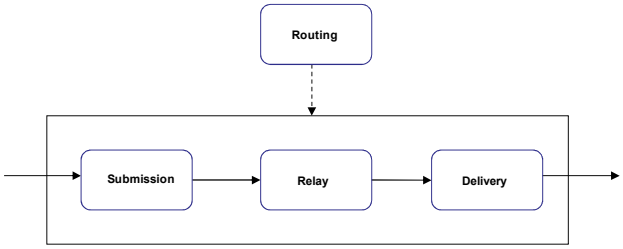


Figure 27 : Décomposition de l'évènement Routing

Il existe une particularité en ce qui concerne l'évènement *Retrieving*. En fonction du mode de transmission, cet évènement peut être basé sur deux mécanismes de transmission différents (décrits dans IMA, chapitre 4.4) : le mode *push* et le mode *pull*. Dans le cas d'un mode *push*, le message est transmis à l'aide du protocole SMTP jusqu'au MS puis le MS transmet le message au MUA destinataire à partir du mécanisme IDLE [149] (extension du standard IMAP). Le destinataire n'a pas besoin de procéder à la récupération du message. Dans le cas d'un mode *pull*, le message est transmis à l'aide du protocole SMTP jusqu'au MS et mis en boite aux lettres. Ensuite, le MUA destinataire procède à la récupération du message à l'aide du protocole IMAP. Le modèle présenté dans ce rapport se base sur le mode *pull*.

L'évènement *Relay* est composé d'un sous-ensemble constitué des évènements *InitTransfer*, *Transfer* et *FinalTransfer*. Cette décomposition d'évènements (cf. Figure 28) permet de réaliser des échanges de messages au sein d'une même ADMD (échange intra-ADMD) ou entre plusieurs ADMDs (échange inter-ADMD). Ces échanges inter-ADMD sont possibles grâce à l'évènement *Transfer* qui permet les échanges entre serveurs BMTA.

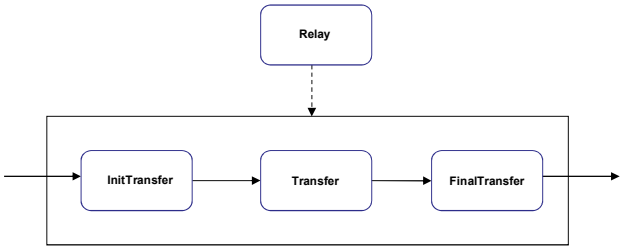


Figure 28 : Décomposition de l'évènement Relay

De manière synthétique, le cycle de vie d'un message est constitué des étapes suivantes :

- Creation (Création)
- Transmission (Acheminement)
  - Submission (Soumission)
  - Relay (Relayage)
    - InitTransfer (Transfert initial)
    - (Transfer)* (Transfert)
    - FinalTransfer (Transfert final)
  - Delivery (Remise)

    - Retrieving (Récupération)
- Display (Affichage)

(*Transfer*)* représente l'occurrence de zéro ou plusieurs évènements *Transfer*. Dans le reste de ce document, nous prenons comme convention que dans le cas intra-ADMD, nous n'avons aucune occurrence de l'évènement *Transfer*, alors que dans le cas inter-ADMD et de la présence de BMTA, nous avons au moins trois occurrences de l'évènement *Transfer*.

Une première description du cycle de vie a été développé à l'aide du langage IOA [150] (*Input/Output Automata*). Elle est fournie en annexe IV.

Il est important de noter que dans certains cas, le cycle de vie peut ne pas dérouler la totalité des évènements et s'arrêter prématurément. C'est le cas par exemple des messages rejetés en raison d'une non-conformité à une politique (anti-virus, anti-spam…) ou d'un dysfonctionnement du système. Cet arrêt prématuré devrait entrainer la génération d'un nouveau message sous la forme d'une notification à destination de l'émetteur du message. Ce type de message prend en général la forme d'un DSN (Delivery Status Notification)[32][31]. Ce nouveau message ne suit pas nécessairement le même cycle de vie que les messages émis par un utilisateur. Par exemple, un DSN émis par le MTA en charge du transfert initial déroulera un cycle de vie limité n'incluant pas de soumission. Nous ne traiterons pas plus en détail dans cette thèse les aspects du cycle de vie des notifications.

6.4.2.2 *Décomposition globale*

La Figure 29 propose une décomposition globale des évènements du cycle de vie d'un message dans un environnement inter-ADMD.

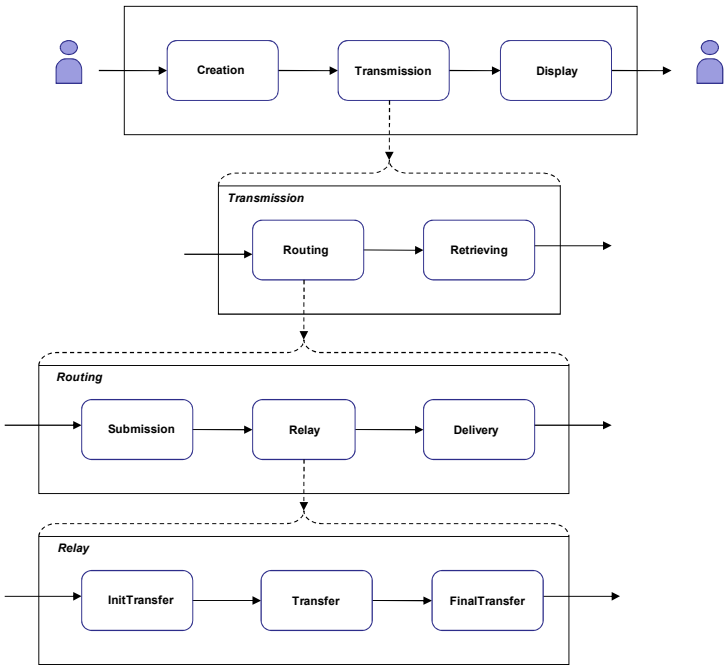


Figure 29 : Décomposition générale du cycle de vie d'un message

6.4.2.3 *Exemple d'un cycle de vie*

L'exemple décrit dans la Figure 30 illustre les concepts de cycle de vie complets et incomplets. Un utilisateur A envoie un message à trois destinataires B, C et D. Le destinataire B récupère le message et l'affiche. Le destinataire C ne récupère pas le message et le destinataire D est inconnu. La correspondance est composée de trois séquences de cycle de vie (CV). Les évènements génériques sont représentés de la manière suivante :

- C : creation
- S : submission
- I : initial transfer
- T : transfer
- F : final transfer
- D : delivery
- R : retrieving
- P : display

Le CV (c1, p1) est complet car il aboutit à un évènement d'affichage p1. Les CV (c1, d1') et (c1, d1'') sont incomplets car ils ne comprennent pas d'évènement d'affichage.

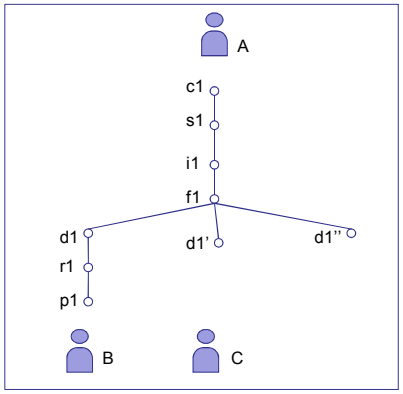


Figure 30 : Exemple d'un cycle de vie composé de séquences complètes et incomplètes

6.4.2.4 *Cycle de vie du message dans un environnement intra-ADMD*

Le cycle de vie d'un message peut être limité à un acheminement dans un environnement intra-ADMD. Dans ce cas, il n'y aura pas d'évènement *Transfer*. L'évènement *InitTransfer* transmet le message directement à l'évènement *FinalTransfer*. Cet exemple est décrit dans la Figure 31.

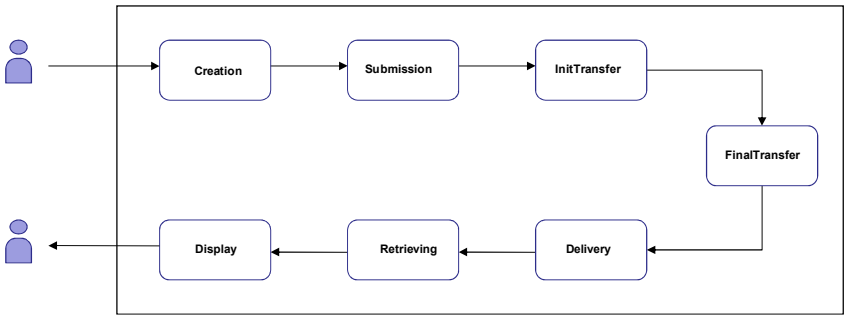


Figure 31 : Cycle de vie d'un message dans un environnement intra-ADMD

6.4.2.5 *Cycle de vie du message dans un environnement inter-ADMD*

A la différence d'un cycle de vie dans un environnement intra-ADMD, un cycle de vie dans un environnement inter-ADMD peut comporter un à plusieurs évènements *Transfer*.

Un exemple permet de décrire en détail la transmission d'un message dans un tel environnement. Dans notre exemple, Alice, en tant qu'auteur, crée un message racine et l'émet à destination de Bob, Charly, Dave et Mark. Alice et Bob appartiennent au domaine d'administration D1, Charly et Dave appartiennent au même domaine D2 et Mark appartient au domaine D3. Nous reprenons dans cet exemple (illustré dans la Figure 32), les évènement génériques précédemment décrits.
Une numérotation est associée à chaque évènement. Par exemple, C1 correspond à l'évènement de création du message par Alice dans le domaine D1. Les évènements d'affichage du message reçu par les destinataires sont mentionnés de la manière suivante; P1 pour Bob, P2' pour Charly, P2" pour Dave et P3 pour Mark.

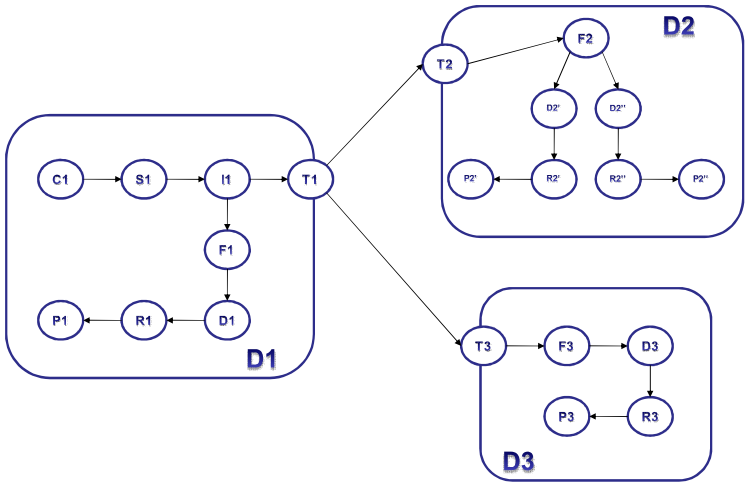


Figure 32: Exemple d'un cycle de vie d'un message dans un environnement inter-ADMD

Dans cet exemple, nous pouvons distinguer quatre séquences de cycle de vie du message: CV(C1, P1), CV(C1, P2'), CV(C1, P2''), CV(C1, P3).

Les travaux de cette thèse limiteront la définition du modèle de correspondance à un environnement intra-ADMD.

6.4.3 Description du modèle de correspondance

Il est possible de décrire le modèle de correspondance de la manière suivante. Soit :

- *C* , une correspondance composée de 1 à *n* correspondances élémentaires *c*,
- *c* , une correspondance élémentaire de *C*,
- *s* , l'émetteur du message,
- *R* , l'ensemble des destinataires. *R* peut être composé de 1 à *n* destinataires *r*,
- et *t* le contexte récupéré par l'agent en charge de l'évènement.

$$C = \{ c_1 .. c_n \}$$

$$R = \{ r_1 .. r_n \}$$

La correspondance *C* est composée de 1 à *n* correspondances élémentaires *c*. Il y a autant de correspondances élémentaires que de destinataires du message.

$$C_{(s,R)} = \{ c_{(s,r_1)} .. c_{(s,r_n)} \}$$

Une correspondance élémentaire est une formalisation du cycle de vie d'un message, elle est donc composée d'une succession d'évènements. Dans notre modèle, chaque évènement a en entrée ou en sortie, un objet que nous appellerons *eo*. Cependant, l'évènement *creation* a en entrée un objet de type *uic* (ensuite transformé en objet *eo)* et l'évènement *display* a en sortie un objet de type *uic*. Dans une correspondance élémentaire, l'objet *eo* subit des transformations durant son cycle de vie.

La correspondance élémentaire c ayant pour émetteur s et destinataire r, et t comme contexte, peut être décrite de la manière suivante :

$$
\begin{aligned}
c_{(s,r)} = \; (\; & creation \; {}^{(t)}_{(uic,eo_{cr})} \; . \\
& submission \; {}^{(t)}_{(eo_{cr},eo_{sb})} \; . \\
& initialTransfer \; {}^{(t)}_{(eo_{sb},eo_{it})} \; . \\
& finalTransfer \; {}^{(t)}_{(eo_{it},eo_{ft})} \; . \\
& delivery \; {}^{(t)}_{(eo_{ft},eo_{dl})} \; . \\
& retrieving \; {}^{(t)}_{(eo_{dl},eo_{rt})} \; . \\
& display \; {}^{(t)}_{(eo_{rt},uic)} \;)
\end{aligned}
$$

L'opérateur "." permet d'exprimer la succession ordonnée d'évènements exécutés dans une correspondance.

Un exemple, illustré dans la Figure 33, permet de décrire ces différents types de correspondance. L'utilisateur A transmet un premier message racine aux utilisateurs E et F. Cette correspondance $C_{(A,\ \{E,F\})}$ est constituée de deux correspondances élémentaires c_1 et c_2. L'utilisateur F renvoie un message sous la forme d'une réponse à l'utilisateur A. Cette correspondance $C_{(F,A)}$ est constituée d'une seule correspondance élémentaire c_4. Ensuite, l'utilisateur F transmet le message émis par A (la correspondance c_2) vers l'utilisateur G. Cette correspondance $C_{(F,\ G)}$ est constituée d'une seule correspondance élémentaire c_5.

L'utilisateur A transmet un nouveau message racine à l'utilisateur G. Cette correspondance, notée $C_{(A,\ G)}$, est constituée d'une seule correspondance élémentaire c_3.

Dans cet exemple, il existe deux correspondances globales, *CG1* et *CG2*. *CG1* est composée des correspondances $C_{(A,\ \{E,F\})}$, $C_{(F,\ A)}$ et $C_{(F,\ G)}$. *CG2* est composée de la correspondance $C_{(A,\ G)}$.

Il est important de noter que l'utilisateur n'a pas connaissance de tous les messages d'une correspondance globale. Dans l'exemple, l'utilisateur A n'a pas connaissance de la correspondance $C_{(F,\ G)}$ qui se déroule entre les utilisateurs F et G.

L'ensemble de tous les messages émis et reçus par un utilisateur est appelé correspondance de l'utilisateur et noté $CU_{(user)}$ avec *user* comme utilisateur. En se basant sur l'exemple, $CU_{(A)}$ est composée des correspondances élémentaires c_1, c_2, c_3 et c_4.

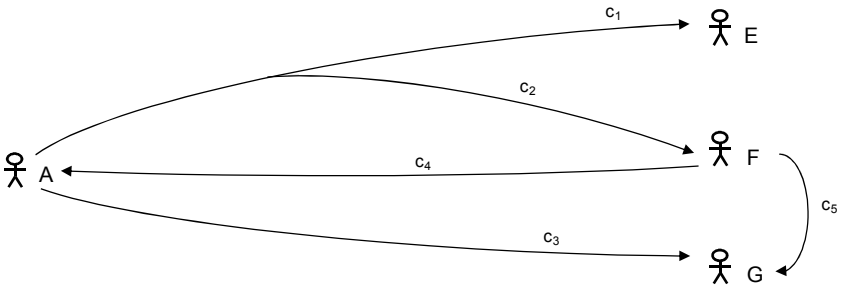


Figure 33 : exemple illustrant les différents types de correspondances

6.5 *Les évènements*

6.5.1 Localisation des évènements dans les agents du système de messagerie

Afin d'avoir une vision précise de l'emplacement des évènements du cycle de vie d'un message dans un système de messagerie électronique, il est intéressant de les associer aux composants décrits dans le document de description des architectures de messagerie IMA [6]. Nous pouvons remarquer que deux évènements sont internes à un unique composant : *creation* et *display*. Il faut cependant noter que l'évènement *creation* se déroule dans le MUA de l'émetteur et *display* se déroule dans le MUA du destinataire. Les évènements en charge de l'acheminement des messages sont positionnés sur deux composants. La Figure 34 propose une vue permettant d'illustrer cette association des évènements avec les composants d'un système de messagerie dans le cas d'un échange intra-ADMD.

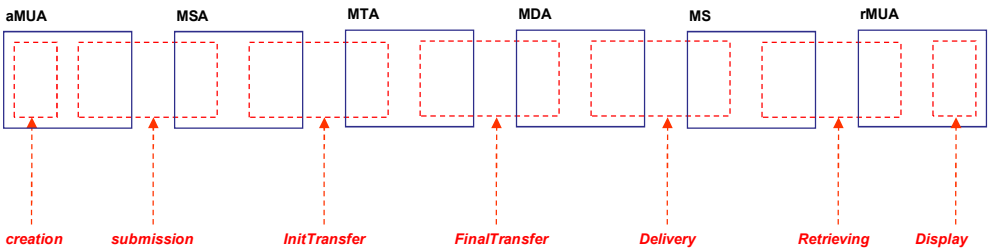


Figure 34 : Association des évènements du cycle de vie avec les composants d'un système de messagerie électronique

6.5.2 L'évènement de création du message

L'évènement qui consiste à créer un message (*creation*) est précédé d'une phase pendant laquelle l'utilisateur envisage la création et l'envoi d'un message. Cette phase correspond à l'intention de communication de l'utilisateur (*uic*). Ensuite, il procède à l'initialisation de la création du message qui correspond à la saisie, dans un formulaire proposé par le client de messagerie, des données du futur message (désignée *form* dans la Figure 35). A ce niveau, le contexte peut être pris en compte. Lors de cette phase, une politique est appliquée.

Ensuite, une phase de génération est exécutée (désignée *generation* dans la Figure 35). Elle consiste à générer le message à partir du contexte et des données fournies par l'utilisateur. Des

informations nécessaires à l'enveloppe de soumission sont également ajoutées. Un objet eo_{cr} est créé. Il correspond à la sortie de l'évènement *Creation*. Ceci est illustré dans la Figure 35.

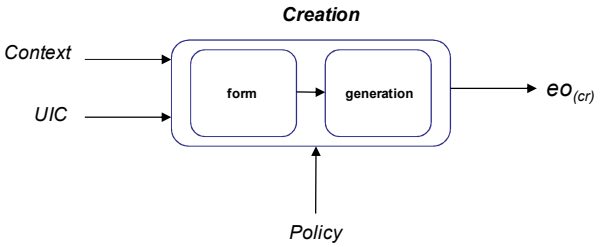


Figure 35 : Evènement de création du message

Pour cet évènement, le contexte peut prendre plusieurs formes. Il peut être composé de la localisation de l'équipement en charge de la création, de l'heure ou encore du type de client de messagerie. L'*uic*, de son côté, est composée des informations nécessaires à la création du message, à savoir les attributs génériques, les attributs spécifiques et le contenu du message. L'objet eo_{cr} peut ensuite poursuivre le cycle de vie du message et être soumis.

6.5.3 **Les évènements de routage du message**

Les évènements *submission*, *initTransfer*, *finalTransfer* et *delivery* composent l'évènement *Routing*. Les processus internes présents dans ces évènements sont similaires. Nous proposons une description générique de ces évènements. La Figure 36 illustre cette description.

Un évènement prend en entrée un objet eo_{in} ainsi que le contexte. Un première phase (illustrée par T dans la Figure 36) est déroulée. Elle correspond au transfert du message entre l'agent client SMTP et l'agent serveur SMTP. Lors de ce transfert, une politique est appliquée. Le transfert de l'objet eo_{in} aboutit à un objet $eo_{post\text{-}in}$. Les données présentes dans l'objet $eo_{post\text{-}in}$ peuvent différer de celles présentes dans l'objet eo_{in} (au niveau de l'enveloppe), en fonction de la négociation SMTP réalisée entre les deux agents.

Ensuite, l'objet $eo_{post\text{-}in}$ entre dans une phase de traitement du message (illustrée par M dans la Figure 36). Lors de ce traitement, une politique est appliquée. La sortie de cette phase est un objet eo_{out}. Cet objet correspond à l'objet en sortie de l'évènement.

Nous appellerons :

- l'objet eo_{in} , le message en entrée
- l'objet $eo_{post\text{-}in}$, le message intermédiaire
- l'objet eo_{out} , le message en sortie

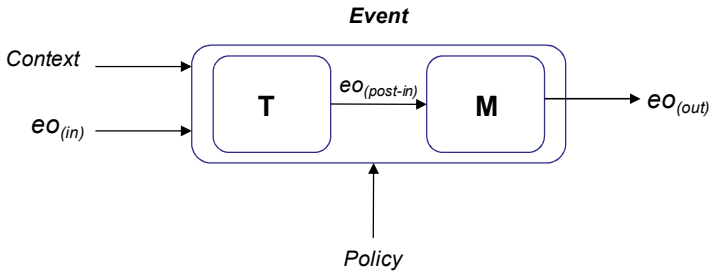


Figure 36 : Evènement générique de routage du message

Le tableau suivant décrit les noms des objets en fonction des évènements présents dans *Routing*.

Evènements	Message en entrée	Message intermédiaire	Message en sortie
submission	eo_{cr}	$eo_{post\text{-}cr}$	eo_{sb}
initTransfer	eo_{sb}	$eo_{post\text{-}sb}$	eo_{it}
finalTransfer	eo_{it}	$eo_{post\text{-}it}$	eo_{ft}
delivery	eo_{ft}	$eo_{post\text{-}ft}$	eo_{dl}

L'évènement *delivery* est différent des trois autres évènements. Il réalise le retrait des informations de l'enveloppe présente dans l'objet eo_{ft}. En effet, l'objet eo_{dl} correspond au message remis dans la boite aux lettres du destinataire et ne contient pas d'information nécessaire à l'acheminement.

6.5.4 **L'évènement de récupération du message**

L'évènement de récupération du message (*retrieving*) prend en entrée un objet eo_{dl} ainsi que le contexte et génère un objet eo_{rt} .

Un première phase (illustrée par T dans la Figure 37) est déroulée. Elle correspond au transfert du message entre l'agent client IMAP et l'agent serveur IMAP. Lors de ce transfert, une politique est appliquée. Le transfert de l'objet eo_{dl} aboutit à un objet $eo_{post\text{-}dl}$. Ensuite, l'objet $eo_{post\text{-}dl}$ entre dans une phase de traitement du message (illustrée par M dans la Figure 37). Lors de ce traitement, une politique est appliquée. La sortie de cette phase est un objet eo_{rt}. Cet objet correspond à l'objet en sortie de l'évènement.

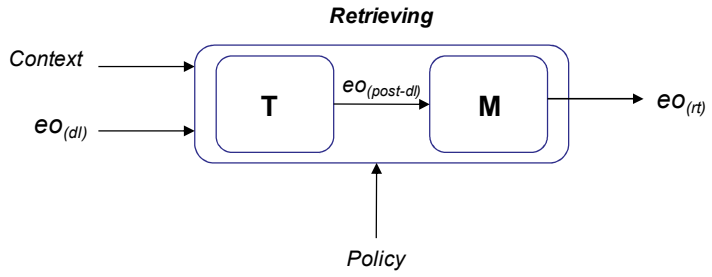


Figure 37 : Evènement de récupération du message

6.5.5 **L'évènement d'affichage du message**

L'évènement d'affichage du message (*display*) prend en entrée un objet eo_{rt} et le contexte. Il se déroule dans le composant MUA. Une première phase *verification* consiste à vérifier la conformité du message. Ensuite, le message est affiché à l'utilisateur à partir du formulaire approprié de visualisation (désignée *form* dans la Figure 38) présent dans le client de messagerie. Lors de ces différentes phases, des politiques sont appliquées. La sortie de l'évènement *display* correspond à l'UIC de l'émetteur.

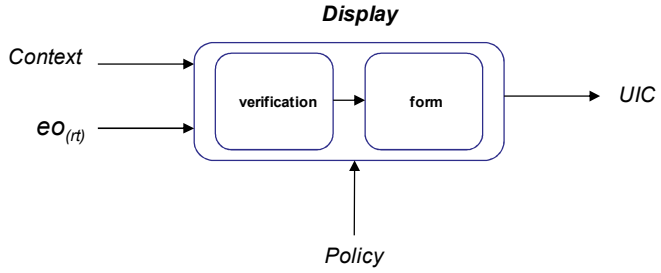


Figure 38 : Evènement d'affichage du message

6.6 *Les politiques de correspondances électroniques*

Le modèle de correspondance électronique permet pour un usage et un contexte donnés de décrire un type de correspondance qui embarque l'ensemble exacte des politiques à appliquer. Avec ce modèle, il devient possible de définir des systèmes de messagerie avancés qui pourront appliquer des politiques appropriées, en fonction de l'usage et du contexte plutôt que des politiques monolithiques sans distinction des échanges. Dans la suite de ce rapport, nous définissons ce type de politique appropriée à l'usage et au contexte, **politique de correspondance**. Ces politiques de correspondance viennent étendre les politiques déjà présentes et décrites dans les standards SMTP[8], SUBMISSION [7], IMF[9] et IMAP[28].

Le concept de politique de correspondance est composé de plusieurs parties distinctes, à savoir un modèle de politique, la découverte de la politique et enfin son application. Les concepts de politique de correspondance exposés ici sont limités à un périmètre intra-domaine.

6.6.1 **Le modèle de politique**

Le modèle de politique a pour objectif de fournir un cadre permettant de spécifier les règles et contraintes, ceci pour un usage et un contexte donnés. Lorsque le message est acheminé au sein du système de messagerie, chaque agent du système est susceptible d'appliquer des règles présentes dans la politique de correspondance.

Une politique de correspondance réunit l'ensemble des contraintes et des règles qui seront appliquées sur le message et lors de son acheminement en fonction d'un usage et d'un contexte donnés. Une politique de correspondance peut être créée à partir d'une recommandation, d'un ensemble d'exigences ou bien issue d'une analyse du besoin des utilisateurs. Dans notre modèle, une politique de correspondance est décrite par un utilisateur autorisé (profil administrateur) pour un domaine puis mise à disposition des utilisateurs.

A partir de ces concepts, nous proposons de définir **une correspondance élémentaire comme réussie quand elle est réalisée, conforme et que les politiques de correspondance appliquées en fonction de l'usage et du contexte sont satisfaites.**

La politique de correspondance (*PC*) à appliquer lors de l'acheminement du message est positionnée par l'émetteur lors de la création du message. Les *PC* sont réunies au sein d'une politique de correspondance de domaine. Le concept de domaine est à associer avec celui d'ADMD présenté dans le chapitre consacré à l'état de l'art de la messagerie électronique.

Le modèle suivant décrit la politique de correspondance de domaine *PD* comme un ensemble de *n* politiques de correspondance PC_1 à PC_n. Dans le cas présent et comme cela a été stipulé précédemment, le modèle est décrit dans un cadre intra-domaine.

$$PD = \{PC_1 .. PC_n\}$$

Chaque agent en charge d'un évènement du cycle de vie du message est susceptible d'appliquer une partie d'une *PC*. Pour cela, une *PC* est décomposée en **politiques de correspondances évènementielles** qui sont associées chacune à un évènement. Nous pouvons retrouver autant de politiques de correspondances évènementielles que d'évènements présents dans le cycle de vie du message. Le modèle suivant décrit une politique de correspondance *PC* comme un ensemble de politiques de correspondance évènementielles.

$$PC = \{p_{cr}, p_{sb}, p_{it}, p_{ft}, p_{dl}, p_{rt}, p_{dp}\}$$

avec

- p_{cr} : politique applicable à l'évènement de *creation*
- p_{sb} : politique applicable à l'évènement de *submission*
- p_{it} : politique applicable à l'évènement de *initTransfer*
- p_{ft} : politique applicable à l'évènement de *finalTransfer*
- p_{dl} : politique applicable à l'évènement de *delivery*
- p_{rt} : politique applicable à l'évènement de *retrieving*
- p_{dp} : politique applicable à l'évènement de *display*

Certaines politiques de correspondance évènementielles ne s'appliquent que sur le message, à savoir p_{cr} et p_{dp} . Les autres politiques de correspondance évènementielles peuvent s'appliquer sur le message électronique et sur sa transmission.

Afin de garantir l'identification d'une politique de correspondance et d'éviter des incohérences, la présence d'une **référence de politique de correspondance** est indispensable. La référence d'une politique de correspondance est composée d'un nom de correspondance, d'une version et du nom de domaine. Cette structure est identique à celle de la référence de correspondance électronique.

6.6.2 Description de la politique de correspondance

Une description des liens entre les différentes politiques est proposée dans le schéma de la Figure 39. Elle représente une vue globale des différents types de politiques de correspondance. La politique de domaine réunit toutes les politiques de correspondance du domaine.

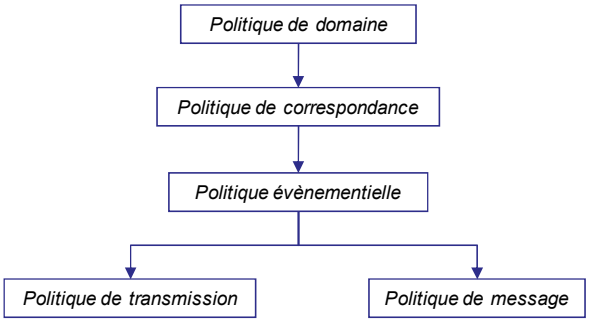


Figure 39 : organisation des différents types de politiques de correspondance

La politique de correspondance évènementielle est composée de deux parties. Une partie dédiée au message et une partie dédiée à sa transmission. La politique de transmission est appliquée lors du transfert du message entre un agent client et un agent serveur. La politique de message est appliquée sur le message, après réception de celui-ci par l'agent serveur. Ce principe d'application de politique est décrit dans le chapitre 6.5 relatif aux évènements.

Afin de décrire les politiques de correspondance, nous utilisons un langage simplifié basé sur XML. Ce choix est justifié par l'extensibilité de ce langage et par le nombre de solutions de développement logiciels existantes.

```
1 <CORRESPONDENCE_POLICY name="aaaaaa" version = "bbbbb"  domain="cccccc">
2    <EVENT eventId="SUBMISSION">
3          <POLICY_TARGET policyTargetId ="TRANSFER">
4               < POLICY_TYPE policyType="TRANSFORMATION">
```

La partie ci-dessus de politique peut être interprétée de la manière suivante :

- la ligne 1 comporte la référence de politique de correspondance, à savoir la composition de *name*, *version* et *domain*.
- la ligne 2 comporte l'évènement concerné du cycle de vie.

- la ligne 3 comporte la cible sur laquelle la politique sera appliquée, à savoir le transfert ou le message.
- la ligne 4 comporte le nom de l'étape concernée, à savoir vérification ou transformation. Ce point est décrit dans le chapitre consacré aux politiques applicables aux messages.

L'unité de base de la politique de correspondance est la règle (*rule*). Elle est composée d'une action, de conditions et de résultats entrainant un effet. Ceci est illustré dans la Figure 40. L'action contient le nom de l'opération à exécuter. Elle peut comporter des paramètres. Les conditions contiennent les critères à satisfaire pour que l'opération puissent s'exécuter. Une condition prend la forme d'un prédicat dans le sens où son évaluation peut renvoyer un résultat vrai ou faux. L'ensemble des conditions renvoie un succès (vrai) ou un échec (faux). Dans le cas d'un échec, l'opération ne peut pas être exécutée. Une règle ne comporte pas nécessairement de condition. Dans ce cas, l'opération est exécutée sans condition. L'exécution de l'action entraine le retour d'une réponse. En fonction de celle-ci, de nouveaux traitements peuvent être réalisés (arrêt de l'application de la politique avec rejet d'un message, avertissement, archivage du message, log...). Le résultat d'une action peut également entrainer l'application de contraintes (chiffrement des flux, authentification du client...). Le contexte peut être pris en compte et traité dans les conditions.

L'expression d'une règle est proposée ci-dessous.

```
5               <RULE ruleId = "dddd">
6                   <ACTION> operation
7                       <PARAMETERS>
8                           <PARAMETER name="pppppp" > PPPPP </PARAMETER>
9                       </PARAMETERS>
10                  </ACTION>
```

Cette description peut être interprétée de la manière suivante :

- la ligne 5 comporte l'identifiant de la règle.
- la ligne 6 comporte le nom de l'opération à exécuter.
- la ligne 8 comporte un paramètre lié à l'opération exécutée.

Ce langage rudimentaire ne permet pas d'enchainement de règles. Des exemples de politiques de correspondance sont fournis dans le chapitre 7.

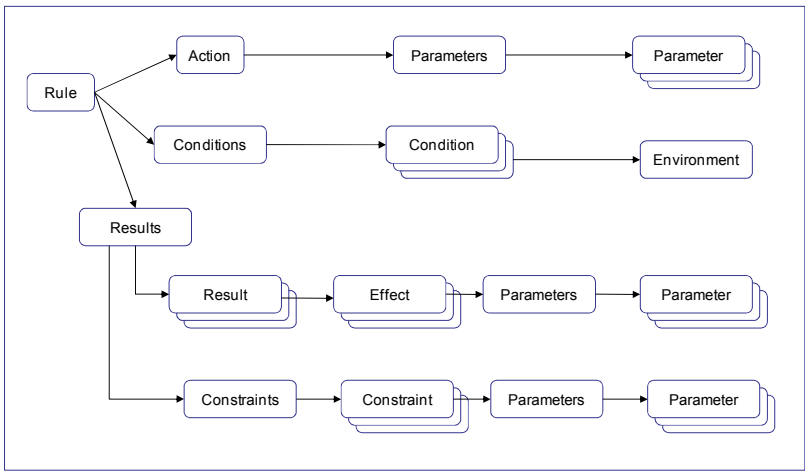


Figure 40 : composition d'une règle

6.6.3 Les politiques de correspondance applicables aux évènements d'acheminement des messages

Les standards SMTP, SUBMISSION et IMAP et certaines extensions SMTP (STARTTLS[84], SMTP AUTH [66]...) définissent les règles et contraintes de conformité garantissant une interopérabilité minimale. Les politiques de correspondance applicables aux évènements d'acheminement des messages, que nous appellerons *PT*, permettent d'étendre ces politiques.

Une *PT* peut être décrite pour chaque évènement du cycle de vie du message, excepté les évènements *creation* et *display*. Ci-dessous, quelques exemples d'actions qui pourraient être spécifiées dans une *PT*:

- *userAuthentication*
- *peerEntityAuthentication*

Un exemple de *PT* est proposé dans le chapitre 7.2 consacré aux exemples de politiques de correspondance.

6.6.4 Les politiques de correspondance applicables aux messages

Les standards IMF et SUBMISSION décrivent des politiques définissant les règles et contraintes de conformité garantissant une interopérabilité minimale. Les politiques de correspondance applicables aux messages permettent d'étendre ces politiques. Nous ne traiterons pas les aspects de politiques anti-spam ou anti-virus dans nos travaux mais le concept des politiques de correspondance que nous proposons pourrait ultérieurement prendre en compte ces aspects.

Nous appellerons *PM*, la politique de correspondance évènementielle dédiée aux messages. Une *PM* est composée de deux étapes qui se déroulent successivement :

- *verification*. Cette étape consiste à appliquer les règles lors de l'étape de vérification du message.
- *transformation*. Cette étape consiste à appliquer les règles lors de la création du message dans le MUA (évènement *creation*). Cette étape consiste également à appliquer des règles, dans les autres évènements du cycle de vie, qui pourront entrainer une modification du message.

Grâce à la présence de ces deux étapes, une *PM* permet de vérifier que les messages reçus respectent la politique de correspondance, de réaliser une éventuelle transformation du message (cachet serveur, horodatage...) et d'exécuter des traitements associés (traces des messages dans des fichiers de logs, archivage...). Une *PM* peut être décrite pour chaque évènement du cycle de vie du message.

Ci-dessous, quelques exemples d'actions qui pourraient être spécifiées dans une *PM*:

- *checkHeaderFieldExist*
- *checkHeaderFieldNotEmpty*
- *checkHeaderFieldValue*
- *addHeaderField*
- *deleteHeaderField*
- *addHeaderFieldValue*
- *deleteHeaderFieldValue*
- *addCryptographicSignature*
- *deleteCryptographiqueSignature*
- *verifyCryptographiqueSignature*
- *addBodyPart*
- *deleteBodyPart*

Des exemples de *PM* sont proposés dans le chapitre 7 consacré aux exemples de politiques de correspondance.

6.6.5 **Découverte et récupération des politiques de correspondance**

Les agents en charge d'un évènement du cycle de vie d'un message doivent disposer de la politique de correspondance à appliquer. Plusieurs options s'offrent à eux. Les politiques peuvent être installées directement et de manière non automatisée sur chaque agent. Cette méthode a pour inconvénient majeur, le peu d'évolutivité du système. En effet, si une nouvelle politique (ou bien une politique modifiée) doit être déployée, la distribution devient vite problématique. Pour répondre à cela, le système doit offrir des mécanismes de récupération automatique des politiques de correspondance.

La solution que nous proposons pour la découverte et la récupération automatiques des politiques est la suivante.

Dans un premier temps, les politiques doivent être publiées au sein d'une zone de stockage dans un serveur de politiques de correspondance (serveur web, par exemple). Ces politiques sont identifiables grâce à une référence de politique (nom, version et domaine). Ensuite, les agents

doivent disposer d'un mécanisme permettant de découvrir la présence du serveur de politiques. Nous proposons de baser ce mécanisme sur le service de nom du domaine (Domain Name System - DNS[48]). Dans ce cas, un agent transmet une requête auprès du DNS afin d'obtenir le nom du serveur de politiques. Dès l'obtention du nom du serveur de politique, l'agent peut émettre une requête auprès de ce serveur et obtenir la politique de correspondance idoine. Différents cas se présentent. Il pourrait être possible de récupérer l'ensemble des politiques du domaine ou bien la politique évènementielle appropriée en fonction de l'agent demandeur ou encore simplement quelques politiques diffusables à l'ensemble des utilisateurs du domaine. Toutefois, ceci pose des problèmes de sécurité. La découverte puis la récupération de certaines politiques pourraient présenter des risques. Ces aspects sécurité ne sont pas traités dans cette thèse et devraient faire l'objet de futurs travaux.

La récupération de politique peut être réalisée pendant le cycle de vie d'un message ou bien en dehors d'un cycle de vie. Par exemple, un agent de type MUA peut récupérer l'ensemble des politiques en fonction des droits liés à un utilisateur sans nécessairement être dans un processus de création d'un message. Un MSA pourrait récupérer une politique pendant le traitement d'un message. Pour des raisons de performances, il est cependant préférable que les agents récupèrent les politiques avant le traitement de messages.

Lors de la récupération de la politique de correspondance, deux cas pourraient se présenter. Soit l'agent récupère la politique globale (comportant la description de la politique pour tous les évènements), soit l'agent récupère la politique évènementielle appropriée. Il serait intéressant de limiter la récupération à la politique évènementielle qui sera appliquée par l'agent.

Un mécanisme alternatif de découverte des politiques a été étudié dans le cadre de cette thèse. Il permet à un MUA de récupérer auprès d'un MSA les politiques de correspondance pour lesquelles un utilisateur a des droits d'utilisation. Nous parlerons dans ce cas, de **capacités de correspondance**. Ce mécanisme nécessite une extension du standard SUBMISSION. Il est décrit en détail dans le chapitre 8 consacré à l'architecture.

6.6.6 **L'application de la politique**

Afin d'appliquer les politiques de correspondance, chaque agent du système de messagerie doit disposer d'un module d'exécution de ces politiques. Dans le cadre de ces travaux, nous décrivons un nouveau composant, appelé CEA (*Correspondence Enforcement Agent*), en charge de l'application de la politique. Dans un système idéal, chaque agent devrait intégrer un CEA. Cependant, certains systèmes pourraient limiter le déploiement de CEA uniquement dans les MUA. Dans ce cas, les agents n'intégrant pas de CEA traiteraient le message sans tenir compte de la correspondance. Cela peut être intéressant lorsque le besoin est limité à l'application de politique sur les agents des utilisateurs finaux.

Le CEA est en charge d'appliquer la politique de correspondance évènementielle appropriée en fonction de l'évènement et de l'agent auquel il est rattaché. Par exemple, le CEA intégré au MSA appliquera les politiques évènementielles de soumission et de transfert initial.

[48] Le Domain Name System (ou DNS, système de noms de domaine) est un service qui permet de traduire un nom de domaine en informations de plusieurs types qui y sont associées, notamment en adresses IP de la machine portant ce nom.

Comme cela a été décrit dans le modèle de correspondance, la politique peut être appliquée lors du transfert d'un message entre un agent client et un agent serveur. La politique peut également être appliquée sur le message lors de sa création, de son affichage ou bien après la réception de celui-ci par l'agent serveur.

En ce qui concerne l'application de la politique de correspondance, elle ne peut être effectuée que sous certaines conditions :

A. l'agent doit disposer d'un CEA
B. l'agent doit être capable de détecter la correspondance courante
C. l'agent doit disposer de la politique à appliquer

6.6.6.1 *Application de la politique lors du transfert du message*

Lors du transfert d'un message, la connaissance de la politique par l'agent serveur est un pré-requis à son application. Cependant, le protocole SMTP ne permet pas le transport de cette information. Afin de rendre possible cette négociation de politique, nous avons étendu le standard SMTP. Cette extension permet à l'agent client de transmettre à l'agent serveur, la référence de la politique de correspondance à appliquer. L'agent serveur peut ensuite appliquer la politique lors du transfert du message. Cela, permet notamment de ne pas débuter ou d'interrompre le transfert d'un message dans le cas où certains critères l'imposent. Cette négociation de politique impose cependant une authentification préalable de l'utilisateur afin de vérifier les autorisations d'utilisation des politiques de correspondance.

L'extension du standard SMTP permettant la négociation de politique de correspondance est décrite dans le chapitre 8.4.

Plusieurs cas peuvent se présenter:

- L'agent client transmet la référence de politique de correspondance et l'agent serveur dispose de cette politique. Dans ce cas, la politique est appliquée.
- L'agent client transmet la référence de politique de correspondance mais l'agent serveur ne dispose pas de la politique de correspondance. Dans ce cas, l'agent serveur doit se charger de la récupération de la politique. Pour différentes raisons, il est possible que l'agent ne puisse pas récupérer la politique. Dans ce cas, le message pourrait être rejeté ou transmis sous certaines conditions.
- L'agent client ne transmet pas de référence de politique de correspondance. Dans ce cas, la politique par défaut sera exécutée.

Quel que soit le cas réalisé, l'agent serveur peut néanmoins appliquer une politique de correspondance tel que cela est décrit dans le chapitre suivant.

6.6.6.2 *Application de la politique sur le message*

Lorsqu'un agent du système de messagerie reçoit un message, il doit identifier la politique de correspondance à appliquer. Pour cela, le message doit comporter une référence de correspondance électronique qui doit être identique à une référence de politique de

correspondance. Cette référence est positionnée dans le message par l'agent en charge de l'évènement de création.

Plusieurs cas peuvent se présenter:

- Le message comporte une référence de correspondance et l'agent dispose de la politique de correspondance associée. Dans ce cas, la politique est appliquée.
- Le message comporte une référence de correspondance mais l'agent ne dispose pas de la politique de correspondance associée. Dans ce cas, l'agent doit se charger de la récupération de la politique. Pour différentes raisons, il est possible que l'agent ne puisse pas récupérer la politique. Dans ce cas, le message pourrait être rejeté ou transmis sous certaines conditions.
- Le message ne comporte pas de référence de correspondance. Dans ce cas, la politique par défaut sera exécutée.

Afin de rendre possible cette négociation de politique, nous avons étendu le standard IMF. Cette extension permet de positionner une référence de politique de correspondance dans le message. Cette extension du standard IMF est décrite dans le chapitre 8.5.

6.6.6.3 *Compatibilité des politiques de correspondances*

Les aspects liés à la compatibilité entre le langage de description des politiques et le CEA sont importants. Le CEA doit être capable d'analyser et d'exécuter les actions présentes dans les politiques de correspondance. Cependant, si le langage de description des politiques évolue, il y a un risque d'incompatibilité. Dans le cas d'incompatibilité, le CEA doit rejeter le message ou appliquer une politique par défaut.

Les aspects de compatibilité ne sont pas traités dans cette thèse et pourraient faire l'objet de travaux futurs.

6.6.6.4 *Autorisation d'utilisation d'usage*

L'application de politique de correspondance devrait être associée à des autorisations d'utilisation d'usage. En effet, l'usage de certains types de correspondance pourrait être limité à un ensemble plus ou moins restreint d'utilisateurs. Pour cela, nous avons intégré des mécanismes de contrôle d'autorisation dans le CEA. Ceci est décrit dans le chapitre 8.2.2 relatif à l'architecture.

6.7 ***Bilan***

Dans ce chapitre, nous avons introduit le concept de correspondance électronique qui offre une solution originale au problème de l'établissement de la réussite d'une communication électronique. Nous proposons de qualifier la réussite d'une communication dans un système de messagerie comme la bonne compréhension par le destinataire de l'intention de communication de l'émetteur. Nous associons l'intention de communication à un type d'échange explicitement choisi par l'émetteur. Une communication sera réussie si le destinataire détermine l'intention de l'émetteur (sous la forme d'un type de message) et que les politiques appliquées lors de la

transmission et l'affichage du message sont satisfaites. Dans le cas contraire, la communication sera non réussie. Ces concepts représentent la base de nos travaux.

Nous définissons une correspondance comme un modèle abstrait qui permet de définir et d'appliquer des politiques appropriées sur le message et lors de sa transmission pour chaque usage et en fonction du contexte courant de l'émetteur. Une correspondance est constituée d'une ou plusieurs séquences de cycle de vie. Nous proposons également une terminologie concernant les différents cas rencontrés.

Le modèle de correspondance électronique proposé permet pour un usage et un contexte donnés de décrire un type de correspondance qui embarque l'ensemble exacte des politiques à appliquer. Avec ce modèle, il devient possible de définir des systèmes de messagerie avancés qui pourront appliquer des politiques appropriées, en fonction de l'usage et du contexte plutôt que des politiques monolithiques sans distinction des échanges. Nous définissons ce type de politique appropriée à l'usage et au contexte, **politique de correspondance**. Ces politiques de correspondance viennent étendre les politiques déjà présentes et décrites dans les standards SMTP, SUBMISSION, IMF et IMAP.

Le cycle de vie d'un message peut être limité à un acheminement dans un environnement intra-ADMD. Dans ce cas, l'application de politiques de correspondance est simplifiée. Il y a une seule autorité administrative et tous les composants du système de messagerie sont sous la responsabilité de cette autorité.

Dans le cas d'un cycle de vie dans un environnement inter-ADMD, les messages sont échangés entre différentes ADMD. L'émetteur et le destinataire du message n'appartiennent pas à la même ADMD et n'appliquent pas nécessairement les mêmes politiques de correspondance car chaque ADMD dispose d'un ensemble indépendant de politiques. Cependant, des relations de confiance et des arrangements complexes peuvent être établis entre ADMD, permettant ainsi l'application de politiques de correspondance compréhensibles et acceptées par les utilisateurs ou composants des ADMD concernées.

L'analyse du cas inter-ADMD nécessite comme préalable la maitrise conceptuelle de l'intra-ADMD. Le périmètre d'investigation concernant ce dernier cas comporte la description d'une architecture représentative ainsi que la définition des extensions protocolaires associées. Une preuve de concept s'avère nécessaire pour évaluer la pertinence ainsi que la maturité des notions proposées. L'importance et la complexité des développements impliqués dans son traitement ainsi que l'homogénéité des concepts proposés nous conduisent à nous focaliser dans cette thèse sur le cas intra-ADMD. Toutefois, tous les concepts qui seront introduits dans nos travaux devront anticiper les investigations futures concernant le cas inter-ADMD.

Chapitre 7

Exemples de politiques de correspondance électronique

Ce chapitre propose deux exemples de description de politiques de correspondance. En effet, les conditions de réalisation de cette thèse imposent de proposer des concepts contribuant à la définition de futurs systèmes de messagerie civile et militaire. Pour le premier exemple, nous nous appuierons sur les spécifications du référentiel général de sécurité appelé RGS [5] dont le but est de fixer les règles que doivent respecter certaines fonctions contribuant à la sécurité de l'information (lors de l'échange d'information entre autorités administratives...). Pour le second exemple, nous prendrons en compte les différentes spécifications existantes pour la définition d'un système de messagerie militaire, compatible avec les exigences de l'OTAN.

7.1 ***Politique RGS basée sur un profil de signature cryptographique***

Ce premier exemple de politique de correspondance est basé sur un profil particulier de signature cryptographique. Comme cela a été abordé dans l'état de l'art, le processus de génération d'une signature électronique met en œuvre deux principaux mécanismes cryptographiques :

- le hachage des données à sécuriser produisant un condensat
- le chiffrement du condensat avec la clé privée du signataire

La signature cryptographique est accompagnée d'un profil de dimensionnement cryptographique qui consiste à décrire le type d'algorithme de hachage, l'algorithme de signature... En général, afin de garantir une interopérabilité avec le plus grand nombre de clients, le profil intègre des algorithmes toujours en vigueur mais relativement faibles. Dans l'exemple présent, nous baserons la politique de correspondance sur le RGS[49]. Le RGS [5] a pour objectif de définir les règles applicables aux échanges entre les usagers et les autorités administratives et entre les autorités administratives. Ce référentiel fixe, selon le niveau de sécurité requis, les règles que doivent respecter certaines fonctions contribuant à la sécurité des informations, parmi lesquelles la signature électronique, l'authentification, la confidentialité ou encore l'horodatage. Le RGS est publié par l'ANSSI [50].

[49] Le Référentiel général de sécurité (RGS) est créé par l'article 9 de l'ordonnance n° 2005-1516 du 8 décembre 2005 relative aux échanges électroniques entre les usagers et les autorités administratives et entre les autorités administratives. Ses conditions d'élaboration, d'approbation, de modification et de publication sont fixées par le décret n° 2010-112 du 2 février 2010 pris pour l'application des articles 9, 10 et 12 de l'ordonnance citée relatif à la sécurité des informations échangées par voie électronique.

[50] Agence nationale de la sécurité des systèmes d'information

L'application du RGS dans un système de messagerie électronique permettrait de respecter les exigences liées aux mécanismes cryptographiques. Il serait ainsi possible, d'imposer des algorithmes particuliers pour les mécanismes de signature et/ou de chiffrement. Le RGS, dans son annexe B1 [151], recommande entre autres, l'utilisation de l'algorithme de hachage SHA-256[152]. Il devient intéressant de décrire dans une politique de correspondance, certaines exigences du RGS. L'exemple proposé dans ce chapitre décrit une politique comportant le type de signature et l'algorithme de hachage à utiliser. Cet exemple ne reprend pas toutes les exigences du RGS. Cependant, la description d'une politique intégrant d'autres exigences du RGS est possible.

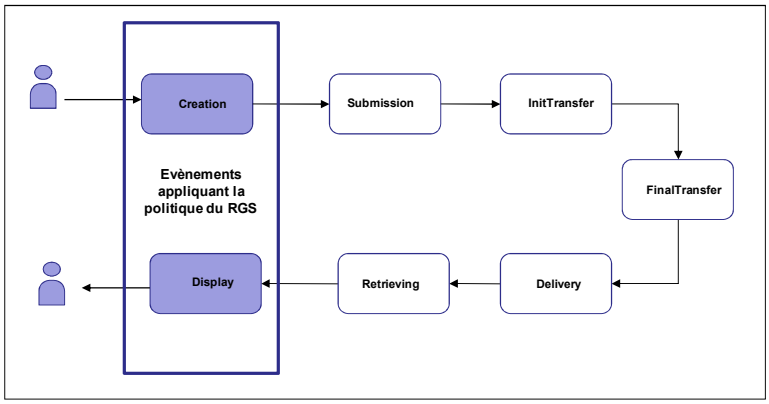


Figure 41 : périmètre d'application de la politique de correspondance du RGS

Dans cet exemple, la politique de correspondance décrit des règles qui ne s'appliquent que sur le message. Deux évènements sont concernés par l'application de la politique de correspondance du RGS. Le périmètre est illustré dans la Figure 41. Pour l'évènement de création du message, la politique comporte une partie *transformation* qui décrit l'algorithme cryptographique à mettre en œuvre lors de la génération de la signature. Pour l'évènement d'affichage du message, la politique comporte une partie *verification* décrivant l'algorithme cryptographique imposé. Cette politique est appliquée dans les composants aMUA (MUA émetteur) et rMUA (MUA destinataire), ceci revient à appliquer une politique de correspondance de bout en bout.

La politique de correspondance du RGS peut être exprimée de la manière suivante. Soit la correspondance PC^{RGS} composée de deux politiques de correspondance évènementielles pour les évènements *creation* et *display*.

$$PC^{RGS} = \{ p_{cr}^{RGS} , p_{dp}^{RGS} \}$$

Une description des politiques évènementielles est réalisée en XML. La première description proposée ici comporte les règles à appliquer lors de l'évènement de création du message exécuté dans le aMUA.

```
1 <CORRESPONDENCE_POLICY name="RGS" version = "1.0"  domain="example.org">
2    <EVENT eventId="CREATION">
3          <POLICY_TARGET policyTargetId ="MESSAGE">
```

```
4            < POLICY_TYPE policyType="TRANSFORMATION">
5                <RULE ruleId ="g1">
6                    <ACTION> addCryptographicSignature
7                        <PARAMETERS>
8                            <PARAMETER name="description" > add a cryptographic signature  </PARAMETER>
9                            <PARAMETER name="mechanism" > SMIME   </PARAMETER>
10                           <PARAMETER name="digestAlgorithm" > SHA-256  </PARAMETER>
11                       </PARAMETERS>
12                   </ACTION>
13               </RULE>
14           <POLICY_TYPE>
15       < POLICY_TARGET >
     </EVENT>
</CORRESPONDENCE_POLICY>
```

Cette politique peut être interprétée de la manière suivante :

- la ligne 1 comporte la référence de politique de correspondance, à savoir la composition de *name*, *version* et *domain*.
- la ligne 2 comporte l'évènement concerné (CREATION)
- la ligne 3 comporte la cible sur laquelle la politique sera appliquée, à savoir le message.
- la ligne 4 comporte le nom du type de politique, à savoir TRANSFORMATION.
- la ligne 5 comporte l'identifiant de la règle.
- la ligne 6 comporte l'opération à exécuter (ajout d'une signature cryptographique).
- la ligne 8 comporte une description de l'opération.
- la ligne 9 comporte le nom du protocole de signature électronique à utiliser.
- la ligne 10 comporte l'algorithme de hachage à utiliser.

L'exécution de cette politique va permettre d'ajouter une signature cryptographique basée sur un profil particulier.

La description proposée ci-dessous correspond aux opérations à exécuter lors de l'évènement d'affichage du message réalisé dans le rMUA.

```
1 <CORRESPONDENCE_POLICY name="RGS" version = "1.0"  domain="example.org">
2    <EVENT eventId="DISPLAY">
3        <POLICY_TARGET policyTargetId ="MESSAGE">
4            < POLICY_TYPE policyType="VERIFICATION">
5                <RULE ruleId ="v1">
6                    <ACTION> checkCryptographicSignature
7                        <PARAMETERS>
8                            <PARAMETER name="description" > check a cryptographic signature  </PARAMETER>
9                            <PARAMETER name="mechanism" > SMIME   </PARAMETER>
10                           <PARAMETER name="digestAlgorithm" > SHA-256  </PARAMETER>
11                       </PARAMETERS>
12                   </ACTION>
13               </RULE>
14           <POLICY_TYPE>
15       < POLICY_TARGET >
     </EVENT>
</CORRESPONDENCE_POLICY>
```

Cette politique peut être interprétée de la manière suivante :

- la ligne 1 comporte la référence de politique de correspondance, à savoir la composition de *name*, *version* et *domain*.
- la ligne 2 comporte l'évènement concerné (DISPLAY)

- la ligne 3 comporte la cible sur laquelle la politique sera appliquée, à savoir le message.
- la ligne 4 comporte le nom du type de politique, à savoir VERIFICATION.
- la ligne 5 comporte l'identifiant de la règle.
- la ligne 6 comporte l'opération à exécuter (vérification d'une signature cryptographique).
- la ligne 8 comporte une description de l'opération.
- la ligne 9 comporte le nom du standard de signature électronique utilisé.
- la ligne 10 comporte l'algorithme de hachage utilisé.

L'application de cette politique va permettre de vérifier qu'une signature est valide et qu'elle respecte un profil particulier de signature cryptographique.

La politique évènementielle DISPLAY revient à vérifier ce qui a été appliqué dans la politique évènementielle CREATION. L'utilisation de ces deux politiques de correspondance permet aux utilisateurs de s'assurer que les messages échangés appliquent la politique de correspondance du RGS.

7.2 *Politique applicable à un système de messagerie militaire*

Un autre cas d'utilisation intéressant est celui de la messagerie militaire. L'OTAN a publié, en 2005, un document sous le titre *Future Military Messaging* [2] (FMM), dont l'objectif était de définir les exigences d'un futur système de messagerie militaire et de clarifier la conception d'un tel système. Ce document propose trois niveaux de services (*high, medium et basic grades*) comportant chacun un ensemble d'exigences à respecter. Le niveau *high grade*, qui se rapproche des concepts de messagerie de confiance, définit les mécanismes à mettre en œuvre pour répondre à un besoin d'échanges d'informations critiques et officielles entre les utilisateurs dans un environnement opérationnel militaire. L'information critique est définie comme une information où des vies ou une mission militaire sont mises en péril si le message n'est pas remis au destinataire dans un délai défini. Les informations officielles sont des informations échangées entre organisations militaires. En ce qui concerne les niveaux *medium* et *basic grades*, ils couvrent le périmètre des échanges professionnels voire privés.

Parallèlement à ces travaux OTAN, A. Melnikov a publié une recommandation [153], compatible avec les exigences présentes dans le document FMM, qui décrit l'ensemble des champs requis pour les messages militaires. Ces champs viennent étendre le standard IMF.

A partir de ces différents travaux, il est intéressant de définir une politique de correspondance électronique applicable dans un système de messagerie militaire. L'idée proposée ici est d'extraire quelques exigences obligatoires et de les intégrer dans l'exemple de politique de correspondance.

Les exigences sélectionnées sont les suivantes:

- peer-entity authentication (STARTTLS)
- user authentication (SASL)
- Precedence (IMF)

Ensuite, il est nécessaire d'identifier les évènements concernés et les mécanismes à mettre en œuvre.

- peer-entity authentication. L'évènement concerné est la soumission. Ce mécanisme peut être mis en œuvre à l'aide du protocole STARTTLS[84].
- user authentication. L'évènement concerné est la soumission. Ce mécanisme peut être mis en œuvre à l'aide du protocole SMTP AUTH [66] et SASL [65] en associant un ou plusieurs algorithmes d'authentification.
- Precedence (IMF). Les évènements concernés sont la création et l'affichage. Par sa présence, cette information indique le niveau d'urgence militaire pour les destinataires en action (champ d'entête To). Ce champ étend le standard IMF.

Dans cet exemple, trois évènements sont donc impliqués par l'application de la politique de correspondance: *creation*, *submission* et *display* . La Figure 42 décrit le périmètre d'application de la politique de correspondance militaire que nous appellerons *MIL*. La politique de correspondance *MIL* peut être exprimée de la manière suivante. Elle est composée de trois politiques de correspondance évènementielles.

$$PC^{MIL} = \{ p_{cr}^{MIL} , \ p_{sb}^{MIL} , \ p_{dp}^{MIL} \}$$

La politique de correspondance évènementielle p_{cr}^{MIL}, exécutée lors de l'évènement *creation*, décrit les règles qui ne s'appliquent que sur le message.

La politique de correspondance évènementielle p_{sb}^{MIL}, exécutée lors de l'évènement *submission*, décrit les règles qui s'appliquent lors du transfert du message entre le aMUA et le MSA puis sur le message.

La politique de correspondance évènementielle p_{dp}^{MIL}, exécutée lors de l'évènement *display*, décrit les règles qui ne s'appliquent que sur le message.

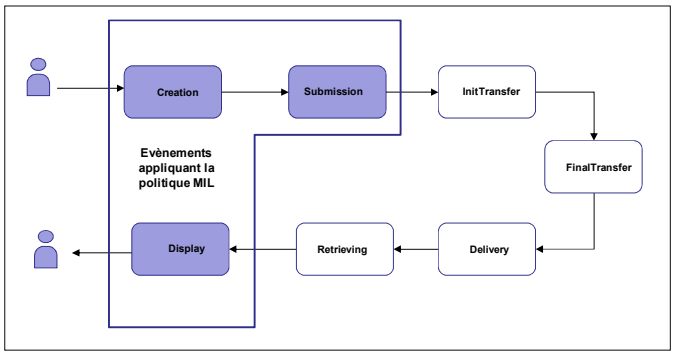


Figure 42 : périmètre d'application de la politique de correspondance de messagerie militaire

7.2.1 **Politique évènementielle applicable lors de l'évènement de création**

La première description proposée ici comporte les règles à appliquer lors de l'évènement de création du message. Pour des soucis de clarté, la description de la politique est réalisée en plusieurs parties.

```
1.1 <CORRESPONDENCE_POLICY name="MIL" version = "1.0"  domain="mil.org">
1.2  <EVENT eventId="CREATION">
1.3       <POLICY_TARGET policyTargetId ="MESSAGE">
1.4            < POLICY_TYPE policyType="TRANSFORMATION">
...
```

La partie ci-dessus de politique peut être interprétée de la manière suivante :

- la ligne 1.1 comporte la référence de politique de correspondance, à savoir la composition de *name*, *version* et *domain*.
- la ligne 1.2 comporte l'évènement concerné (CREATION).
- la ligne 1.3 comporte la cible sur laquelle la politique sera appliquée, à savoir le message.
- la ligne 1.4 comporte le nom du type de politique, à savoir TRANSFORMATION.

```
2.1                <RULE ruleId ="g1">
2.2                     <ACTION> addHeaderField
2.3                          <PARAMETERS>
2.4                               <PARAMETER name="description" > add primary precedence header field   </PARAMETER>
2.5                               <PARAMETER name="headerFieldName" > MMHS-Primary-Precedence  </PARAMETER>
2.5                          </PARAMETERS>
2.7                     </ACTION>
...
```

La partie ci-dessus propose une description d'une règle permettant l'ajout d'un champ d'entête.

- la ligne 2.1 comporte l'identifiant de la règle.
- la ligne 2.2 comporte le nom de l'opération exécutée pour ajouter le champs d'entête.
- la ligne 2.4 comporte une description de l'opération.
- la ligne 2.5 comporte le nom du champ d'entête à créer, à savoir MMHS-PRIMARY-PRECEDENCE. La valeur de ce champ est récupérée à partir du formulaire de création de message.

7.2.2 **Politique évènementielle applicable lors de l'évènement de soumission**

La description proposée ici comporte les règles à appliquer lors de l'évènement de soumission du message.

```
3.1 <CORRESPONDENCE_POLICY name="MIL" version = "1.0"  domain="mil.org">
3.2  <EVENT eventId="SUBMISSION">
3.3       <POLICY_TARGET policyTargetId ="TRANSFER">
3.4            < POLICY_TYPE>
3.5                 <RULE ruleId = "t1">
3.6                      <ACTION> peerEntityAuthentication
3.7                           <PARAMETERS>
3.8                                <PARAMETER name="description" > STARTTLS   </PARAMETER>
3.9                                <PARAMETER name="status"> required </PARAMETER>
3.10                               <PARAMETER name="method"> clientAuthentication </PARAMETER>
3.11                          </PARAMETERS>
3.12                     </ACTION>
...
```

La partie ci-dessus propose une description d'une règle concernant l'authentification mutuelle du MUA et du MSA basée sur le standard STARTTLS.

- la ligne 3.2 comporte l'évènement concerné (SUBMISSION).
- la ligne 3.3 comporte la cible sur laquelle la politique sera appliquée, à savoir le transfert.
- la ligne 3.5 comporte l'identifiant de la règle.
- la ligne 3.6 comporte le nom de l'opération exécutée pour l'authentification mutuelle.
- la ligne 3.8 comporte une description de l'opération.
- la ligne 3.9 comporte le statut de l'opération, à savoir exigé.
- la ligne 3.10 comporte la méthode imposée.

```
4.1                 <RULE ruleId = "t2">
4.2                     <ACTION> userAuthentication
4.3                         <PARAMETERS>
4.4                             <PARAMETER name="description" > SASL user authentication   </PARAMETER>
4.5                             <PARAMETER name="status"> required </PARAMETER>
4.6                             <PARAMETER name="algorithm"> SASL External </PARAMETER>
4.7                         </PARAMETERS>
4.8                     </ACTION>
...
```

La partie ci-dessus propose une description d'une règle concernant l'authentification de l'utilisateur basée sur le standard SASL.

- la ligne 4.1 comporte l'identifiant de la règle.
- la ligne 4.2 comporte le nom de l'opération exécutée pour l'authentification de l'utilisateur.
- la ligne 4.4 comporte une description de l'opération.
- la ligne 4.5 comporte le statut de l'opération, à savoir exigé.
- la ligne 4.6 comporte la méthode d'authentification imposée.

7.2.3 **Politique évènementielle applicable lors de l'évènement d'affichage**

La description proposée ici est liée à l'évènement d'affichage du message et donc appliquée dans le MUA du destinataire.

```
5.1 <CORRESPONDENCE_POLICY name="MIL" version = "1.0"  domain="mil.org">
5.2  <EVENT eventId="DISPLAY">
5.3         <POLICY_TARGET policyTargetId ="MESSAGE">
5.4                 < POLICY_TYPE policyType="VERIFICATION">
...
```

La partie ci-dessus de politique peut être interprétée de la manière suivante :

- la ligne 5.1 comporte la référence de politique de correspondance, à savoir la composition de *name*, *version* et *domain*.
- la ligne 5.2 comporte l'évènement concerné (DISPLAY).
- la ligne 5.3 comporte la cible sur laquelle la politique sera appliquée, à savoir le message.
- la ligne 5.4 comporte le nom du type de politique, à savoir VERIFICATION.

```
6.1          <RULE ruleId ="v1">
6.2              <ACTION> checkHeaderField
6.3                  <PARAMETERS>
6.4                      <PARAMETER name="description" > check primary precedence header field  </PARAMETER>
6.5                      <PARAMETER name="headerFieldName" > MMHS-Primary-Precedence  </PARAMETER>
6.6                  </PARAMETERS>
6.7              </ACTION>
...
```

La partie ci-dessus propose une description d'une règle permettant la vérification d'un champ d'entête.

- la ligne 6.1 comporte l'identifiant de la règle.
- la ligne 6.2 comporte le nom de l'opération exécutée pour vérifier le champs d'entête.
- la ligne 6.4 comporte une description de l'opération.
- la ligne 6.5 comporte le nom du champ d'entête à vérifier, à savoir MMHS-PRIMARY-PRECEDENCE.

Dans cet exemple, la politique DISPLAY permet de vérifier ce qui a été appliqué dans la politique CREATION. Cependant, d'autres scénarios de politique sont envisageables. Il serait par exemple possible de mettre en œuvre des mécanismes de contrôle de rôle. Pour cela, la politique SUBMISSION pourrait être étendue et comporter des règles imposant, une vérification du rôle porté par le signataire du message puis l'ajout d'une contre-signature cryptographique attestant de ce rôle. La politique DISPLAY comporterait des règles visant à vérifier la signature et la contre-signature. Ainsi, la politique DISPLAY serait satisfaite seulement si les politiques CREATION et SUBMISSION ont été correctement appliquées. Ce dernier exemple démontre l'intérêt de la décomposition des politiques en évènements.

7.3 *Bilan*

Dans ce chapitre, nous avons présenté deux exemples de description de politiques de correspondance. En effet, les conditions de réalisation de cette thèse imposaient de proposer des concepts contribuant à la définition de futurs systèmes de messagerie civile et militaire. Pour le premier exemple, nous nous sommes appuyés sur les spécifications du référentiel général de sécurité appelé RGS [5]. Pour le second exemple, nous avons pris en compte les différentes spécifications existantes pour la définition d'un système de messagerie militaire, compatible avec les exigences de l'OTAN.

Chapitre 8

Architecture d'un système de messagerie basé sur le concept de correspondance électronique

Comme cela a été rappelé dans les objectifs de cette thèse, ces travaux doivent aboutir à la description d'une architecture représentative qui prend en compte les spécifications d'un service de messagerie militaire et civile. Pour cela, nous proposons une architecture basée sur le modèle de correspondance électronique. Nous présentons également en détail les services et extensions nécessaires à l'application des concepts de correspondance électronique dans une architecture basée sur les standards.

Le concept de modèle de correspondance électronique peut être appliqué sur le modèle en trois couches du service de messagerie, décrit dans le chapitre sur l'état de l'art, à savoir:

- L'accès au service
- L'acheminement du message
- Le message

Pour la définition de notre architecture, nous distinguons deux périmètres. Ils sont illustrés dans la Figure 43. Le modèle peut s'appliquer sur le message (partie haute de la figure) et/ou sur l'accès au service et l'acheminement du message (partie basse de la figure). Dans la suite des travaux, l'accès au service étant mis en œuvre lors des évènements d'acheminement des messages, il est intégré dans la couche liée à l'acheminement du message. Il est également intéressant de faire apparaitre dans ce même schéma, les évènements du cycle de vie du message avec les composants décrits dans le document d'architecture IMA [6] et d'y faire figurer les objets *eo*.

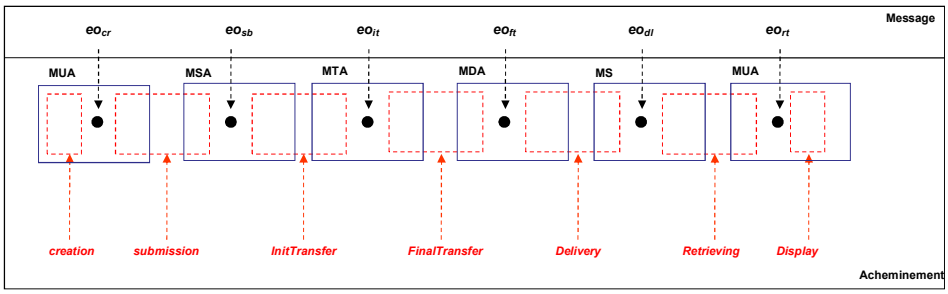


Figure 43 : Périmètres d'application du modèle de correspondance électronique

Il est important de noter que la mise en œuvre des concepts de correspondance électronique ne remet pas en cause le fonctionnement initial d'un système de messagerie.

8.1 *Application du concept de correspondance électronique*

L'application du concept de correspondance électronique dans un système de messagerie impose une évolution de l'architecture standard. Il est important de rappeler que nous limitons l'application du modèle à une architecture intra-ADMD. Le concept de correspondance est basé sur le cycle de vie des messages, lui-même composé d'un ensemble d'évènements. Ces derniers se déroulent au cœur des agents du système de messagerie. Chaque agent de l'architecture doit être capable d'appliquer les politiques définies. La solution que nous proposons est d'intégrer dans chaque agent du système de messagerie, un nouveau composant en charge de l'application des politiques de correspondance. Ce composant est le CEA (*Correspondence Enforcement Agent*). Il se positionne en interface entre l'architecture de messagerie conventionnelle et l'architecture de correspondance électronique. Une vue globale de l'architecture est proposée dans la Figure 44. Grâce à ce principe d'architecture, le fonctionnement initial du service de messagerie n'est pas remis en cause.

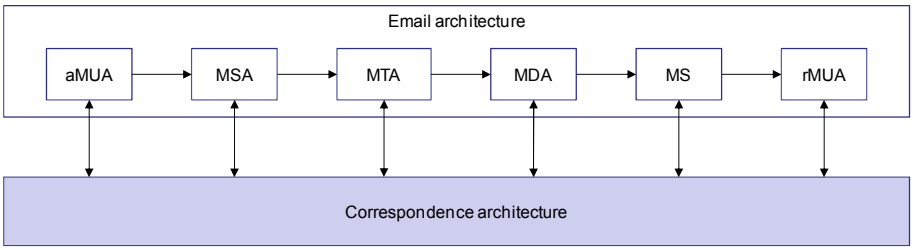


Figure 44 : Architecture de messagerie intégrant les composants en charge de la correspondance électronique

Le CEA interagit avec un autre composant, le CR (*Correspondence Policies Repository*) en charge du stockage des politiques de correspondance. Dans le principe proposé d'architecture, le CEA récupère la politique et en extrait les actions. Elles sont exécutées à l'aide d'API[51] embarquées dans le CEA et dans l'agent du système de messagerie. Il serait intéressant, à terme, de disposer d'un CEA embarquant toutes les API. Cela permettrait de limiter les risques d'incohérence entre les API présentes dans l'agent et les API présentes dans le CEA.

L'architecture générique d'un agent embarquant un CEA est décrite dans la Figure 45. Nous pouvons remarquer la présence de deux composants internes, C1 et C2. Ces composants sont en charge de l'interface entre l'architecture de messagerie traditionnelle et l'architecture de correspondance Ils sont associés chacun à un évènement. Lors de la réception de l'objet eo_n, C1 interagit avec le CEA afin de permettre l'application de la politique de correspondance appropriée. A la fin des traitements, l'objet $eo_{n'}$ est créé et ensuite traité par C2 qui interagit avec le CEA, de la même manière que C1. L'objet issu des traitements dans C2 est $eo_{n''}$.

[51] En informatique, une interface de programmation (souvent désignée par le terme API pour Application Programming Interface) est un ensemble normalisé de classes, de méthodes ou de fonctions qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels.

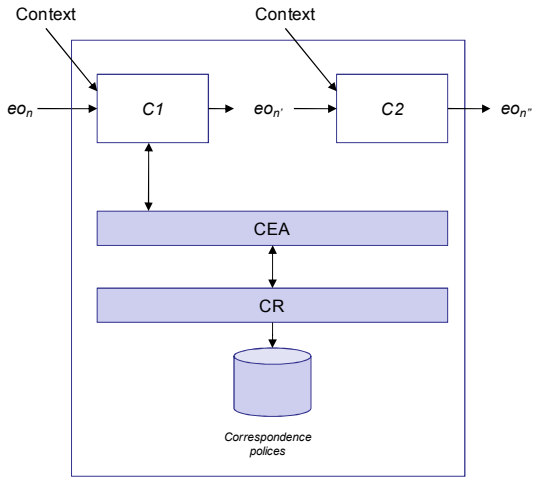


Figure 45 : Architecture interne générique d'un agent du système de messagerie embarquant un CEA

8.2 *Description des agents du système*

8.2.1 **Description détaillée du MUA**

Une description détaillée du MUA permet de visualiser les interactions entre les composants internes. Cette description est proposée dans la Figure 46 pour la partie émission (aMUA) et dans la Figure 47 pour la partie réception (rMUA).

Les interactions dans le aMUA sont les suivantes. Le composant en charge de la création du message reçoit les informations de l'*uic* et du contexte. Les informations de l'*uic* proviennent de formulaires appropriés que l'utilisateur peut utiliser. Le composant en charge de la création interagit avec le CEA afin de récupérer la politique applicable pour l'évènement *creation*. Le CEA récupère la politique auprès du CR, composant en charge du stockage interne des politiques. Ensuite, le CEA applique la politique de correspondance sur le message. Il peut s'appuyer sur des API présentes dans le aMUA ou bien embarquées dans le CEA. Le message créé prend la forme de l'objet eo_{cr}. Ce message est transmis au composant en charge de la soumission qui interagit avec le composant présent dans l'agent serveur. Afin de négocier les services à mettre en œuvre lors de la soumission du message, une extension du protocole SUBMISSION a été définie. Cette extension est présentée dans le chapitre 8.4.

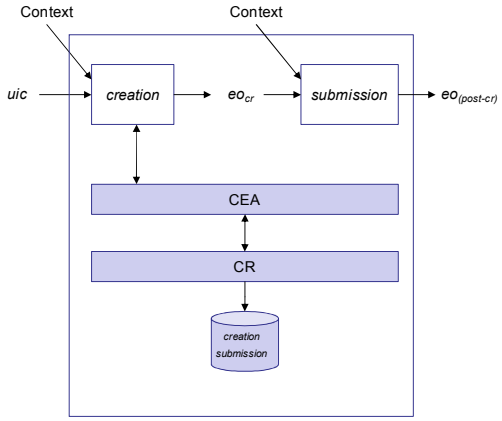


Figure 46 : description détaillée du aMUA

Pour la partie réception, les interactions dans le rMUA sont les suivantes. Le composant en charge de la récupération du message reçoit l'objet $eo_{(post\text{-}dl)}$. Ce composant interagit avec le CEA afin de récupérer la politique applicable pour l'évènement *retrieving*. Le CEA récupère la politique auprès du CR puis applique la politique de correspondance sur le message qui prend la forme de l'objet eo_{rt}. Ce message est transmis au composant en charge de l'affichage. Ce dernier reçoit les informations du contexte. Le composant en charge de l'affichage interagit avec le CEA afin de récupérer la politique liée à l'évènement *display*. Le CEA récupère la politique auprès du CR et l'applique sur le message.

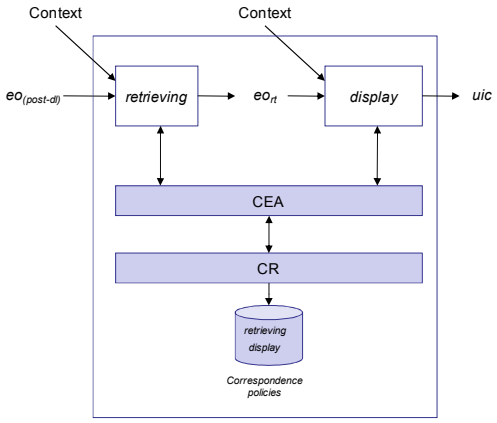


Figure 47 : description détaillée du rMUA

8.2.2 **Description détaillée du MSA**

Le MSA est un composant important du système de messagerie. Il représente l'interface entre le MUA et le système. Hutzler et al. [29] rappellent que le lien le plus faible dans la chaine de confiance est la soumission du message. D. Crocker [6], de son côté, souligne que lors de la soumission d'un message, le MSA est le composant en charge du transfert de responsabilité entre le l'utilisateur et le système de messagerie. Afin de renforcer les services du MSA, l'architecture interne que nous proposons intègre trois services additionnels :

- un mécanisme d'application de la politique de correspondance

- un contrôle d'accès basé sur des autorisations de correspondance
- un mécanisme de transmission des capacités de correspondance vers chaque utilisateur autorisé

Le premier service correspond au mécanisme d'application de politique tel que décrit dans le paragraphe 6.6.6.

Le deuxième service permet un contrôle d'accès en fonction des autorisations de chaque utilisateur. L'objectif est de définir des autorisations par utilisateur et de permettre un contrôle d'accès lors de la soumission d'un message.

Le troisième service est un peu particulier dans le sens où il ne participe pas directement au cycle de vie du message. L'objectif est de proposer un mécanisme de distribution de capacités de politiques de correspondance à destination des utilisateurs autorisés. Grâce à ce mécanisme, les utilisateurs récupèrent leurs capacités de correspondance et peuvent procéder à une installation automatique ou manuelle des politiques. Ces concepts de capacités sont décrits dans le chapitre 8.7.1.

Deux évènements du cycle de vie du message se déroulent dans le MSA; *submission* et *initTransfer*. Nous allons, dans un premier temps, décrire le processus de soumission.

Afin de permettre une négociation de la correspondance courante, nous avons dû étendre le standard SUBMISSION. Cette extension est détaillée dans le chapitre 8.4. Deux cas de figure peuvent se présenter lorsque le MUA initie une session avec le MSA :

1. Le MUA et le MSA supportent l'extension du protocole de soumission. Dans ce cas, une négociation de la correspondance courante ainsi qu'un contrôle d'autorisation sont réalisés avant la transmission du message. Ceci permet notamment la mise en œuvre de services particuliers (chiffrement, authentification...).
2. Le MUA ou le MSA (ou les deux) ne supporte pas l'extension du protocole de soumission. Dans ce cas, il n'y a pas de négociation de la correspondance ni de contrôle d'autorisation avant la transmission du message. La transmission du message se déroule sans traitement particulier et la politique par défaut est appliquée.

La Figure 48 illustre les interactions présentes dans le MSA. Le composant en charge de la soumission du message reçoit l'objet $eo_{(post\text{-}cr)}$ et les informations du contexte. On suppose les informations de contexte données par des composants internes à l'agent. Le composant interagit avec le CEA afin de récupérer la politique applicable pour l'évènement *submission*. Le CEA récupère la politique auprès du CR puis l'applique sur le message et lors de la soumission. Le message récupéré prend la forme de l'objet eo_{sb}. Cet objet est transmis au composant en charge du transfert initial qui ouvre une connexion avec l'agent serveur, puis initie la transaction SMTP pour le transfert du message.

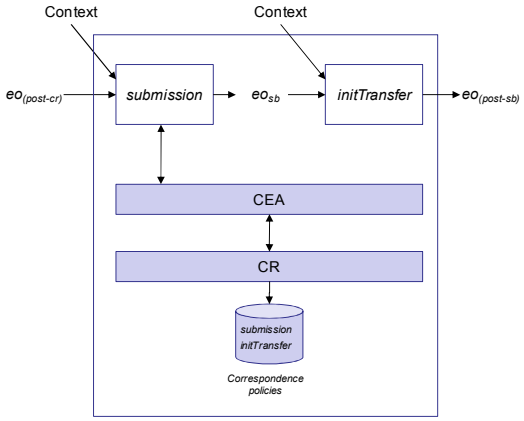


Figure 48 : description détaillée du MSA

Dans le cadre de cette thèse, une architecture particulière concernant le processus de soumission a été définie. Cette architecture intègre un mécanisme de routage basé sur le concept de correspondance L'objectif est de permettre un routage des messages vers un composant ou une architecture particulière. Ce routage est précédé d'une étape de contrôle d'accès basé sur des autorisations. Ce principe d'architecture a fait l'objet d'un dépôt de brevet [154], fourni en annexe III. Il est présenté ci-dessous.

L'architecture, présentée dans la Figure 49, comporte trois nouveaux composants par rapport à un système de messagerie classique :

- Le CEA (*Correspondence Enforcement Agent*) dont le rôle a déjà été décrit dans ce rapport.
- Le CPA (*Correspondence Provider Agent*) en charge des autorisations et des informations de routage des messages.
- Le CR (*Correspondence Repository*) en charge du stockage des informations de politiques.

Le composant PEP (*Policy Enforcement Point*) est issu, pour sa part, d'une architecture d'autorisation décrite dans le RFC 2753 [155].

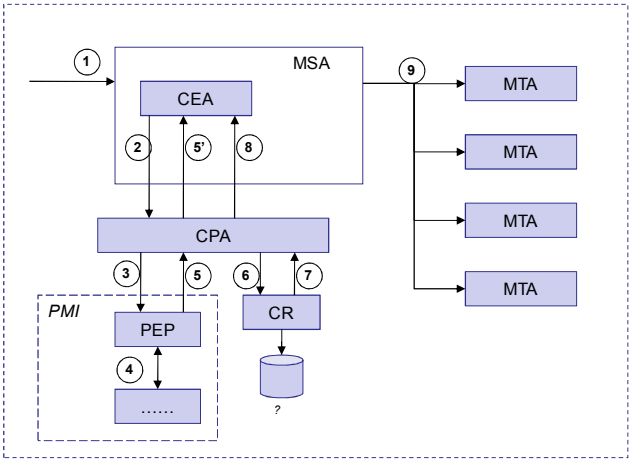


Figure 49 : description détaillée du MSA avec la soumission différenciée

La figure ci-dessus décrit les interactions entre les différents composants. Le CEA se positionne en interface entre le composant CPA et le MSA. Nous pouvons remarquer que le CEA est interne au MSA alors que le CPA et le CR sont des composants externes au MSA.

Lors de la soumission d'un message (1), le CEA émet une requête vers le CPA pour obtenir le statut d'autorisation de la correspondance courante ainsi que l'identifiant du MTA suivant (2). Le CPA est en interface avec le composant CEA, le PEP et le CR. Lors de la réception de la requête en provenance du CEA, le CPA émet une requête vers le PEP (3). Ce dernier obtient le statut d'autorisation auprès de l'architecture d'autorisation (4) puis le transmet vers le CPA (5). Si la réponse est négative, le CPA renvoie l'information au CEA (5') qui la relaie au MSA afin de bloquer la transaction. Si la réponse est positive, le CPA émet une requête de demande d'informations auprès du CR (6). Le CR transmet au CPA les informations nécessaires au transfert différencié (7), dans le cas présent, l'identifiant du MTA suivant. Le CPA transmet l'information au CEA (8) qui la relaie au MSA. Lorsque le MSA reçoit le message en provenance du MUA, il procède à la transmission du message vers le MTA identifié (9).

Grâce à ce principe d'architecture, un seul composant CPA peut être interfacé avec plusieurs agents du système de messagerie (MSA, MTA, MDA...). De plus, cette architecture dédiée apporte une certaine souplesse dans la mise à jour des politiques et dans l'ajout de nouvelles politiques. Un premier prototype a été développé dans le cadre du projet AUDIENCE.

8.2.3 **Description détaillée du MTA**

Une description détaillée du MTA permet de visualiser les interactions entre les composants internes. Cette description est présentée dans la Figure 50.

Les interactions dans le MTA sont les suivantes. Le composant en charge du transfert initial reçoit l'objet $eo_{(post\text{-}sb)}$ en provenance du MSA et les informations du contexte. Ce composant interagit avec le CEA afin de récupérer la politique applicable pour l'évènement *initTransfer*. Le CEA récupère la politique auprès du CR puis l'applique sur le message. Le message transféré prend la forme de l'objet eo_{it}. Ce message est transmis au composant en charge du transfert final qui ouvre une connexion avec l'agent serveur, puis initie la transaction SMTP pour le transfert du message.

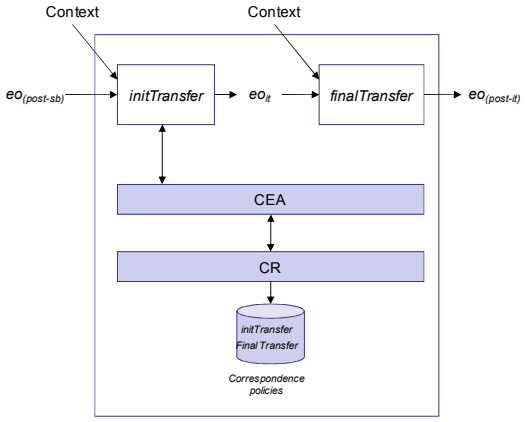


Figure 50 : description détaillée du MTA

8.2.4 **Description détaillée du MDA**

Une description détaillée du MDA permet de visualiser les interactions entre les composants internes. Cette description est présentée dans la Figure 51.

Les interactions dans le MDA sont les suivantes. Le composant en charge du transfert final reçoit l'objet $eo_{(post\text{-}it)}$ en provenance du MTA et les informations du contexte. Ce composant interagit avec le CEA afin de récupérer la politique applicable à l'évènement *finalTransfer*. Le CEA récupère la politique auprès du CR puis l'applique sur le message. Le message transféré prend la forme de l'objet eo_{ft}. Ce message est transmis au composant en charge de la remise qui ouvre une connexion avec l'agent serveur, puis initie la transaction SMTP pour le transfert du message.

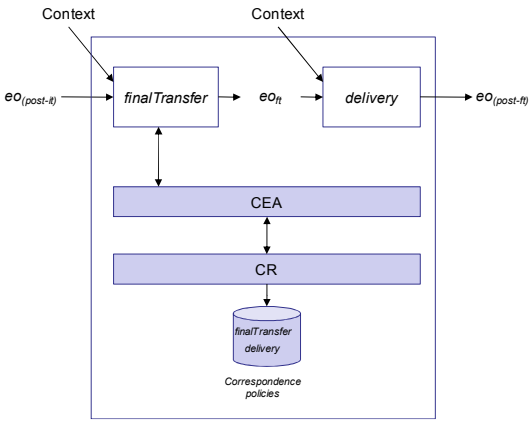


Figure 51 : description détaillée du MDA

8.2.5 **Description détaillée du MS**

Une description détaillée du MS permet de visualiser les interactions entre les composants internes. Cette description est représentée dans la Figure 52. Le principe de fonctionnement du

MS est particulier dans le sens où il agit en tant qu'agent serveur pour l'évènement *delivery* ainsi que pour l'évènement *retrieving*.

Les interactions dans le MS sont les suivantes. Le composant en charge de la remise reçoit l'objet $eo_{(post\text{-}ft)}$ en provenance du MDA et les informations du contexte. Ce composant interagit avec le CEA afin de récupérer la politique applicable à l'évènement *delivery*. Le CEA récupère la politique auprès du CR puis l'applique sur le message. Le message transféré prend la forme de l'objet eo_{dl}. Cet objet correspond au message qui est mis dans la boite aux lettres du destinataire.

Ensuite, le MUA de l'utilisateur émet une requête de récupération auprès du composant en charge de l'évènement *retrieving*. Ce composant reçoit également les informations du contexte dans lequel se trouve le MUA. Le composant interagit avec le CEA afin de récupérer la politique applicable à l'évènement *retrieving*. Le CEA récupère la politique auprès du CR et applique la politique de correspondance sur le message et sur le protocole de récupération.

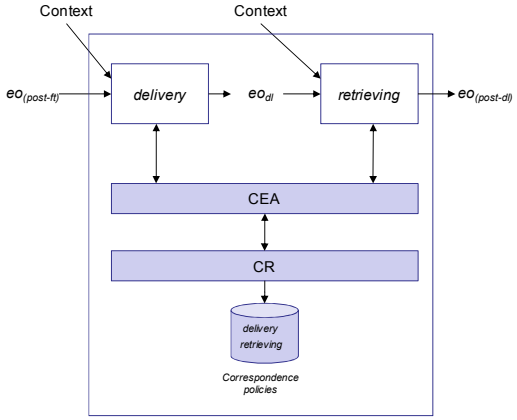


Figure 52 : description détaillée du MS

8.3 *Identification des politiques de correspondances électroniques*

Les agents du système de messagerie doivent être capables d'identifier la politique de correspondance à appliquer lorsque un message est en cours de traitement dans un des évènements du cycle de vie. Cet identifiant de politique doit être unique afin d'éviter toutes incohérences avec d'autres politiques. La solution proposée est d'identifier une correspondance électronique à partir de trois informations:

- le nom de la correspondance
- la version de la correspondance
- le nom de domaine (au sens DNS[52]) du fournisseur de la correspondance

[52] Domain Name System

L'agrégation de ces trois informations (qui prend la forme d'une référence de correspondance électronique) garantit l'unicité de la correspondance, pour un domaine donné. Cette référence doit être présente dans le message et dans le protocole de soumission. A partir de cette référence, l'agent peut déterminer la politique à appliquer. L'ajout de la référence de correspondance dans le message et dans le protocole de soumission nécessite la définition d'extensions des standards concernés. Les chapitres suivants décrivent les extensions des standards SUBMISSION et IMF.

8.4 *Extension du standard SUBMISSION*

Le standard SUBMISSION, dont la description est fournie dans le RFC 6409 [7], est basé sur le standard SMTP. A l'aide de SUBMISSION, il est possible d'appliquer une politique dédiée à l'évènement de soumission. Cependant, il ne permet pas l'application de politiques appropriées en fonction de critères présents dans le concept de correspondance. Pour offrir ce type de service, il est nécessaire d'étendre le standard SUBMISSION. Le principe d'extension est possible et décrit dans SMTP [8]. L'objectif de l'extension du standard SUBMISSION est double. Le premier objectif est de décrire un mécanisme de négociation de la correspondance lors de la soumission d'un message entre un MUA et un MSA. Ceci afin d'appliquer des politiques appropriées. Le second objectif est de récupérer les capacités de correspondance pour un utilisateur donné. Ce dernier point est présenté dans le chapitre 8.7relatif à la découverte des politiques.

Une partie des travaux de cette thèse ont consisté à définir une extension du standard SUBMISSION.

8.4.1 **Négociation de la correspondance lors de la soumission du message**

Le concept de négociation de la correspondance permet à un MUA et un MSA d'échanger des informations sur la correspondance courante. Elle permet de s'assurer que l'utilisateur est autorisé à émettre un message pour une correspondance donnée. Dans le cas où l'utilisateur n'est pas autorisé à émettre, l'envoi du message peut être bloqué. Cela permet notamment de stopper la transaction avant même l'envoi du message et donc de ne pas procéder au transfert de responsabilité.

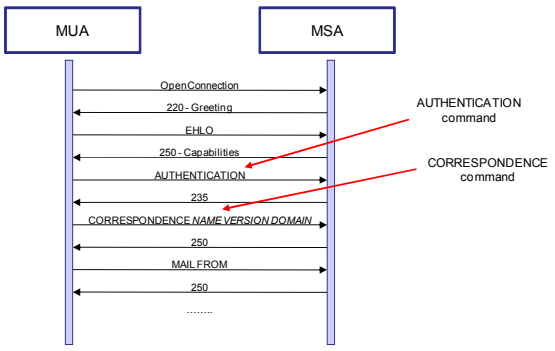


Figure 53 : négociation de la correspondance lors de la soumission d'un message

Le principe de fonctionnement est le suivant. Un nouveau mot clé, à savoir CORRESPONDENCE, est ajouté au protocole SUBMISSION. Le MSA présente le support de cette nouvelle extension en réponse au mot clé EHLO envoyé par le MUA. Le MUA souhaitant utiliser cette extension transmet une commande CORRESPONDENCE avec des arguments.

Une description détaillée est proposée ci-dessous.

1. Le serveur est en attente d'une ouverte de connexion
2. Le MUA ouvre une session avec le MSA sur le port 587
3. le MSA répond à la demande d'ouverture de session
4. le MUA envoie une commande EHLO pour débuter la transaction SUBMISSION
5. le MSA transmet la liste des extensions supportées au MUA
6. l'utilisateur s'authentifie auprès du MSA
7. si l'authentification est négative alors la session est en attente d'une nouvelle authentification ou d'une commande QUIT. Si l'authentification est positive, la session continue
8. la transaction débute lorsque le MUA envoie la commande CORRESPONDENCE avec les informations de la correspondance courante. La commande CORRESPONDENCE est suivie du nom de la correspondance, de la version et du nom de domaine.
9. Un contrôle basé sur des autorisations est réalisé. Si l'utilisateur est autorisé, le MUA reçoit un code positif puis la transaction continue. Dans le cas contraire, le MSA transmet au MUA un échec et en fonction de la politique définie, la transaction continue ou est stoppée. Le MSA valide les informations de la correspondance courante
10. Le MUA et le MSA continuent l'échange conformément au protocole SUBMISSION.

Ces étapes sont illustrées dans la Figure 53 et décrites en détail dans l'exemple fourni dans la Figure 54.

L'envoi de la commande CORRESPONDENCE par le MUA n'est possible qu'après une authentification de l'utilisateur, basée sur l'extension SMTP AUTH [66] et SASL [65]. Un exemple simple d'authentification est proposé dans les échanges 6 et 7 de la Figure 54. La commande CORRESPONDENCE pourrait comporter des paramètres supplémentaires afin de transmettre au MSA, des informations nécessaires à l'application de règles particulières.

Afin de garantir la confidentialité des informations de correspondance transmises avec le mot clé CORRESPONDENCE, la transaction peut être précédée d'une négociation TLS [10] basée sur l'extension STARTTLS [84].

Un utilisateur malveillant pourrait tenter de transmettre des informations de correspondance différentes du type de message à envoyer. Par exemple, il pourrait soumettre un message professionnel avec des informations de correspondance privée (dans le protocole SUBMISSION). Afin de remédier à ce risque, le MSA peut vérifier ces informations après la soumission en procédant à une analyse du message. Dans le cas où les informations de correspondance seraient incohérentes, le message pourrait être rejeté et un administrateur pourrait être informé.

```
1   : <waits for connection on TCP port 587>
2   : <opens connection>
3   : 220 foo.com Simple Mail Transfer Service Ready
4   : EHLO bar.com
5   : 250-foo.com greets bar.com
5   : 250-AUTH GSSAPI DIGEST-MD5 PLAIN
5   : 250-ENHANCEDSTATUSCODES
5   : 250-STARTTLS
5   : 250-CORRESPONDENCE
5   : 250 HELP
6   : AUTH PLAIN dGVzdAB0ZXN0ADEyMzQ=
7   : 235 2.7.0 Authentication successful
8   : CORRESPONDENCE NAME=officiel VERSION=1.0 DOMAIN=example.org
9   : 250 OK
10  : MAIL FROM:<Smith@bar.com>
11  : 250 OK
12  : RCPT TO:<Jones@foo.com>
13  : 250 OK
14  : DATA
15  : 354 Start mail input; end with <CRLF>.<CRLF>
16  : Blah blah blah...
17  : ...etc. etc. etc.
18  : .
19  : 250 OK
20  : QUIT
21  : 221 smtp.example.com Service closing transmission channel
```

Figure 54 : Exemple d'une transaction de soumission intégrant l'envoi d'une commande CORRESPONDENCE

La rédaction d'un document, avec pour objectif une soumission auprès de l'IETF, est en cours.

L'extension décrite dans ce chapitre est limitée au standard SUBMISSION et n'est mise en œuvre qu'entre le aMUA et le MSA. Il est donc nécessaire de disposer d'un autre mécanisme permettant d'identifier la correspondance courante lors de l'acheminement du message. Une possibilité serait d'étendre les standards SMTP et IMAP. Ceci pourrait faire l'objet des travaux ultérieurs. L'autre possibilité est de se baser sur le format du message. C'est ce qui est présenté dans le chapitre suivant.

8.5 *Extension du standard IMF*

Lorsqu'un message est acheminé au sein d'un système de messagerie, il est nécessaire d'identifier la correspondance courante. Pour cela, le message doit comporter des informations relatives à la correspondance. Le principe que nous proposons est d'ajouter, lors de l'évènement de création, des informations dans un champ dédié de l'entête du message. Ceci a pour principal intérêt de permettre la transmission de ces informations jusqu'au destinataire final. La présence de ce champ dans le message permet aux différents agents du système de messagerie de détecter la référence de correspondance et d'appliquer la politique appropriée.

8.5.1 **Champ d'identification de la correspondance électronique**

Le standard IMF permet l'ajout de champs dans l'entête du message. Ces champs peuvent soit faire l'objet d'une standardisation ou bien se limiter à un besoin particulier et ne pas être publiés.

Le RFC 4021 [156] est un document, maintenu par l'IANA, réunissant au sein d'un même référentiel[53], les champs ayant fait l'objet d'une standardisation.

Notre objectif est de définir un nouveau champ permettant le transport de la référence de correspondance électronique. Cette référence doit contenir les informations suivantes:

- le nom de la correspondance
- la version de la correspondance
- le nom de domaine (au sens DNS) du fournisseur de la correspondance

Ces informations sont réunies au sein d'un même champ dont la description ABNF est fournie ci-dessous.

```
correspondence-header = "Correspondence:" cor-name ";" cor-version ";" cor-domain CRLF

cor-name      = [FWS] "name" [FWS] "=" [FWS] %d34 structured-field %d34

cor-version   = [FWS] "version" [FWS] "=" [FWS] %d34 structured-field %d34

cor-domain    = [FWS] "domain" [FWS] "=" [FWS] %d34 structured-field %d34

structured-field = 1*allowed-character

allowed-character =    UTF8-xtra-char              ; cf. RFC5335, I18N Email Headers
```

Un message respectant le standard IMF et porteur de ce champ sera appelé XIMF (*eXtended IMF*). Un exemple permettant d'illustrer la description de ce champ d'entête est fourni ci-dessous.

Correspondence: name="example" ; version="1.02"; domain="example.org"

Les agents du système embarquant les mécanismes décrits dans cette thèse doivent être capables de détecter la présence de ce champ et d'en extraire la référence de correspondance. Dans le cas contraire, cela peut être notamment problématique pour le destinataire. En effet, si le MUA n'intègre pas ces mécanismes, il ne permettra pas un traitement approprié et l'utilisateur ne pourra pas visualiser toutes les informations présentes dans le message étendu. Pour répondre à ce problème, nous proposons la définition d'un type MIME spécifique et dédié aux concepts de correspondance. Lors de la création d'un message, un objet MIME particulier est ajouté dans le corps du message.

Lors de la réception d'un message étendu, deux cas peuvent se présenter :

- soit l'agent n'embarque pas de mécanisme de détection de la référence de correspondance. Il peut cependant afficher un contenu informant l'utilisateur que le message est étendu. Ce contenu peut également comporter des informations comme l'adresse d'un site permettant le téléchargement d'un client compatible.

[53] http://www.iana.org/assignments/message-headers/message-headers.xhtml

- soit l'agent embarque des mécanismes de détection de la référence de correspondance et est capable de traiter le message. Dans ce cas, il masque le contenu MIME spécifique présent dans le message. Ce mécanisme est le même que celui mis en œuvre lors de la réception d'un message avec un contenu MIME encapsulant une signature cryptographique. Le MUA effectue les traitements de vérification de la signature mais n'affiche pas le contenu MIME lié à la signature.

La rédaction d'un document décrivant la référence de correspondance et l'extension du standard IMF, avec pour objectif une soumission auprès de l'IETF, est en cours.

8.5.2 **Extension du message basé sur le concept de messages semi-structurés**

La présence du champ de référence de la correspondance permet à un agent du système de détecter un message étendu (message de type XIMF). Cependant, la présence d'autres éléments de services peut être imposée pour une correspondance donnée. Le message peut comporter d'autres métadonnées dans l'entête du message. Il est par exemple possible d'ajouter un label de sensibilité de l'information ou bien des informations spécifiques à un projet. Le message pourrait également comporter des pièces jointes particulières ou bien une signature cryptographique.

Chaque type de correspondance peut imposer la description de formats spécifiques de message. Afin de prendre en compte ces aspects, nous introduisons la notion de **profil de message** que nous définissons comme **la description des éléments de services qui doivent être présents dans le message pour un type de correspondance donné**. Un élément de service peut prendre la forme d'un champ d'entête particulier, d'une pièce jointe, d'une signature cryptographique, d'une donnée d'horodatage...).

Un exemple de format de message militaire permet de décrire plus en détail ce concept de profil. Dans cet exemple, le profil peut être basé sur le RFC6477 [153] et les travaux [157]. Ces deux documents proposent une liste des champs d'entête à utiliser afin de garantir une interopérabilité avec un système militaire de type MMHS basé sur les protocoles de l'Internet. Les champs définis sont les suivants:

- *MMHS-Exempted-Address*
- *MMHS-Extended-Authorisation-Info*
- *MMHS-Subject-Indicator-Codes*
- *MMHS-Handling-Instructions*
- *MMHS-Message-Instructions*
- *MMHS-Codress-Message-Indicator*
- *MMHS-Originator-Reference*
- *MMHS-Primary-Precedence*
- *MMHS-Copy-Precedence*
- *MMHS-Message-Type*
- *MMHS-Other-Recipients-Indicator-To*
- *MMHS-Other-Recipients-Indicator-CC*
- *MMHS-Acp127-Message-Identifier*
- *MMHS-Originator-PLAD*

Le profil de cet exemple de message militaire, décrit en langage XML simplifié, est proposé dans la Figure 55. Il ne décrit la présence que de deux champs d'entête. Cependant, dans un profil complet, tous les champs devraient être présents.

```
1 <CORRESPONDENCE  name="MIL" version = "1.0"  domain="example.org">
2   <EVENT eventId="CREATION">
3       <MESSAGE_PROFILE>

4           < HEADER_FIELDS>
5               <HEADER_FIELD> MMHS-Extended-Authorisation-Info
6                   <STATUS> Optional </STATUS>
7               </HEADER_FIELD>

8               <HEADER_FIELD> MMHS-Primary-Precedence
9                   <STATUS> Mandatory </STATUS>
10              </HEADER_FIELD>

.....

11          < /HEADER_FIELDS>

12      < /MESSAGE_PROFILE >
13  </EVENT>

14  <EVENT eventId="DISPLAY">
15      <MESSAGE_PROFILE>

16          < HEADER_FIELDS>
17              <HEADER_FIELD> MMHS-Extended-Authorisation-Info
18                  <STATUS> Optional </STATUS>
19              </HEADER_FIELD>

20              <HEADER_FIELD> MMHS-Exempted-Address
21                  <STATUS> Mandatory </STATUS>
22              </HEADER_FIELD>

.....

23          < /HEADER_FIELDS>

24      < /MESSAGE_PROFILE >
25  </EVENT>

26 </CORRESPONDENCE >
```

Figure 55 : exemple de profil d'un message militaire

Nous pouvons remarquer que ce profil fait référence à deux évènements; *creation* et *display*. Ce profil décrit donc le format des messages eo_{cr} et eo_{dp} .

La Figure 56 illustre deux exemples de messages ; IMF et XIMF. Le message XIMF affiché comporte des champs d'entête additionnels qui correspondent aux champs militaires décrits dans le profil précédent.

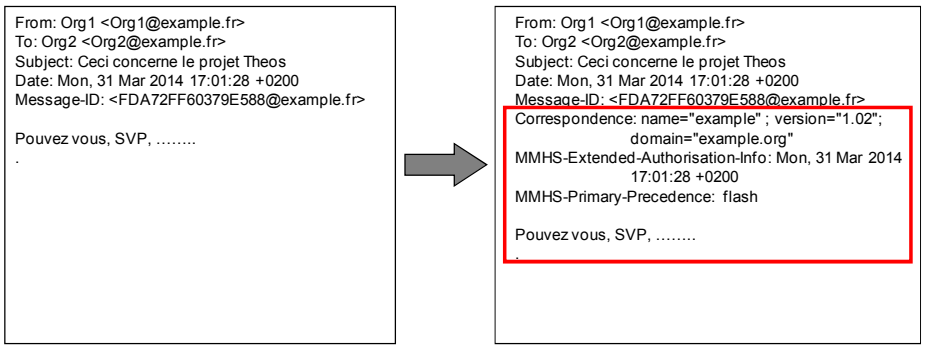


Figure 56 : exemple de message IMF et XIMF

8.6 ***Sécurisation des messages étendus***

Comme nous l'avons décrit dans les chapitres précédents, l'utilisation de messages semi-structurés permet des traitements avancés. De tels messages nécessitent la définition de champs d'entête particuliers, comme la référence de correspondance. Cependant, la sécurisation de ce type de messages impose une extension des standards actuels. Nous avons vu dans l'état de l'art que le standard S/MIME ne garantit pas la sécurisation des champs d'entête présents dans le message. Il est donc nécessaire de disposer d'un mécanisme permettant de sécuriser la totalité des données présentes dans le message étendu. Les travaux de cette thèse ont consisté à définir une extension des standard S/MIME [11] et CMS [78] permettant de garantir la sécurisation des champs présents dans l'entête du message. Grâce à cette technologie, appelée SHF (*Securing Header Fields*), il est possible d'appliquer les propriétés de sécurité sur la totalité du message. Cette extension est décrite en détail dans le document [158].

La technologie SHF est basée sur les principes de champs d'attributs signés présents dans l'objet CMS. Elle impose la description d'une politique de sécurité qui décrit la liste des champs d'entête à sécuriser. Les agents doivent partager la même politique de sécurité. Celle-ci sera intégrée dans la politique de correspondance applicable sur le message. Dans le cas où le destinataire ne dispose pas d'un client capable d'interpréter les champs sécurisés par la technologie SHF alors il les ignore. La technologie SHF ne remet pas en cause l'interopérabilité S/MIME. Une description d'un objet CMS avec une structure SHF est fournie dans la Figure 57.

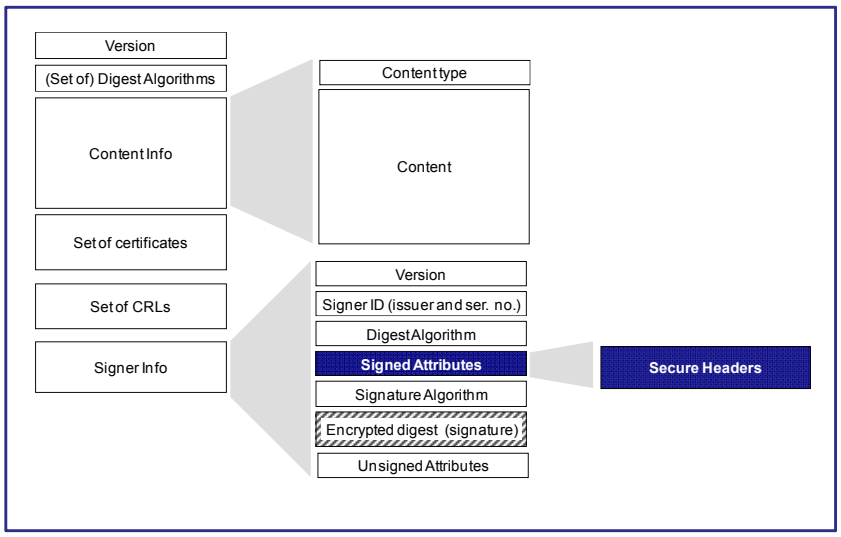


Figure 57 : Objet CMS intégrant une structure SHF

Afin de décrire en détail la technologie SHF, une description en notation ASN.1 est fournie ci-dessous. La structure principale est composée de deux parties.

SecureHeaderFields ::= SET {
canonAlgorithm Algorithm,
secHeaderFields HeaderFields }

canonAlgorithm permet de spécifier le type d'algorithme de canonisation à appliquer. Ce mécanisme est basé sur la spécification DKIM. Il existe deux types d'algorithme de canonisation: *simple* et *relaxed*. L'algorithme *simple* ne tolère aucune modification alors que l'algorithme *relaxed* permet de légères modifications comme le remplacement d'espaces ou encore le reformatage de lignes.

secureHeaderFields contient une copie des champs d'entête présents dans le message. Les champs à sécuriser sont spécifiés dans la politique de sécurité. *secureHeaderFields* est composé de une à plusieurs structures *HeaderField*. Chaque structure *HeaderField* a trois champs: *name*, *value* et *status*. Le champ *HeaderFieldName* contient le nom du champ d'entête, le champ *HeaderFieldValue* contient la valeur du champ d'entête et le champ *HeaderFieldStatus* contient un statut nécessaire pour les processus de chiffrement et de déchiffrement. La structure *HeaderField* est décrite ci-dessous.

HeaderField ::= SEQUENCE {
field-Name HeaderFieldName,
field-Value HeaderFieldValue,
field-Status HeaderFieldStatus DEFAULT duplicated }

8.6.1 Processus de génération et de vérification de la signature électronique

La technologie proposée étend le processus de signature décrit dans CMS. le processus de génération de signature suit les étapes suivantes:

- les champs à sécuriser sont sélectionnés, en fonction de la politique de sécurité.

- l'algorithme de canonisation est appliqué pour chaque champ d'entête sélectionné.
- chaque champ d'entête sélectionné est intégré dans la structure *SecureHeaderFields*.
- le champ concernant l'algorithme est renseigné en fonction de l'algorithme de canonisation configuré.
- la structure *SecureHeaderFields* est encapsulée dans la structure des attributs signés de l'objet *signerInfo*.
- le processus de génération de la signature est exécuté comme décrit dans CMS.

La Figure 58 illustre les étapes décrites ci-dessus (1) et le processus de génération de signature comme décrit dans CMS (2 et 3).

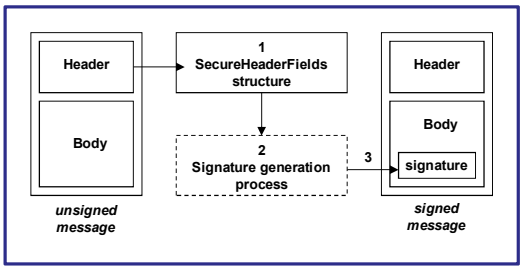


Figure 58 : schéma simplifié du processus de génération de la signature

La vérification SHF est basée sur le processus de vérification de la signature décrit dans CMS. A la fin de ce processus et si la signature est valide, les étapes suivantes sont exécutées:

- le type d'algorithme de canonisation spécifié dans la structure *SecureHeaderFields* est récupéré.
- pour chaque champ présent dans la signature, le processus recherche le champ correspondant dans l'entête du message. Si il n'y a pas d'équivalence alors la signature est invalide.
- l'algorithme de canonisation est appliqué sur chaque valeur de champ d'entête du message.
- pour chaque champ, le processus compare la valeur stockée dans la structure *SecureHeaderFields* avec la valeur retournée par l'algorithme de canonisation. S'il n'y a pas d'équivalence alors la signature est invalide.
- le processus vérifie qu'il n'y a pas de champ ajouté dans l'entête par rapport aux champs définis dans la politique. Si un champ a été ajouté, alors la signature est invalide.
- le processus vérifie que tous les champs présents dans le message et déclarés dans la politiques de sécurité, sont aussi présents dans la structure *SecureHeaderFields*. Si un champ est manquant alors le processus alerte l'utilisateur et indique le nom du champ manquant.

Le client de messagerie affiche les informations concernant la validation de chaque champ d'entête. Le processus de génération de la signature S/MIME intégrant la technologie SHF est

illustré dans la Figure 59. Les champs d'entête du message à sécuriser sont encapsulés dans les attributs signés de l'objet CMS (1) et la signature est générée (2).

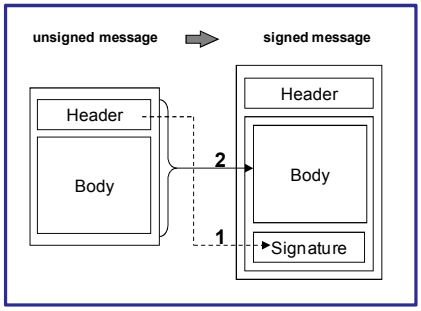


Figure 59 : Processus de génération de signature avec la technologie SHF

8.6.2 **Processus de chiffrement et de déchiffrement**

La caractéristique des processus de chiffrement et de déchiffrement de la technologie SHF est qu'ils ne sont pas réalisés par le MUA. Ceci est principalement dû à une limitation IMAP. Le standard IMAP ne permet pas la modification des messages stockés dans les boites aux lettres. Afin de fournir les services de chiffrement et de déchiffrement, nous avons choisi de développer un nouveau composant basé sur la spécification *Domain Security* [159]. Le composant qui réalise ces processus est appelé *Domain Confidentiality Authority* (DCA).

La Figure 60 illustre les composants d'architecture et une transmission d'un message qui est signé et chiffré. L'architecture est composée de deux entités ADMD. Les concepts d'ADMD sont décrits dans le chapitre relatif à l'état de l'art. Le MUA émetteur (Sender MUA) exécute le processus de génération de signature et transmet le message signé (S). Le DCA de l'ADMD A reçoit le message S et le chiffre (E). Le DCA de l'ADMD B reçoit le message E, le déchiffre (S) et le transmet au MUA destinataire (Recipient MUA) qui procède à la vérification de la signature.

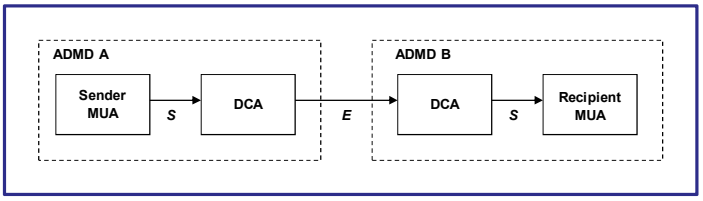


Figure 60 : Architecture basée sur la technologie SHF

Les processus de chiffrement et de déchiffrement ne peuvent être exécutés que dans le cas où le contenu à chiffrer correspond à un objet signature S/MIME et si la structure *SecureHeaderFields* est encapsulée dans les attributs signés de la signature. Ces processus utilisent les champs statuts présents dans *HeaderFieldStatus*. Les trois valeurs de statut sont *duplicated*, *deleted* and *modified*. Le principal intérêt des statuts est qu'il est possible de décrire un comportement particulier par champ d'entête.

Lors du processus de chiffrement :

- si le statut est *duplicated*, le champ ne subit aucune transformation.
- si le statut est *deleted*, le DCA retire le champ du message.
- si le statut est *modified*, le DCA remplace la valeur du champ par une nouvelle valeur.

Lors du processus de déchiffrement :

- si le statut est *duplicated*, le champ ne subit aucune transformation.
- si le statut est *deleted*, le DCA réécrit le champ (nom et valeur) dans le message à partir du champ présent dans la structure *SecureHeaderFields*.
- si le statut est *modified*, le DCA réécrit la valeur du champ présente dans *HeaderFieldValue* de la structure *SecureHeaderFields*.

Le processus de chiffrement S/MIME intégrant la technologie SHF est illustré dans la Figure 61. En fonction des statuts, les champs d'entête du message peuvent être protégés (1) et le message est chiffré (2). Avec cette technologie, il est possible de transférer un message à travers un système non sécurisé. Cependant, les mécanismes de chiffrement et de déchiffrement sont optionnels et la technologie SHF peut se limiter à l'application des mécanismes de signature.

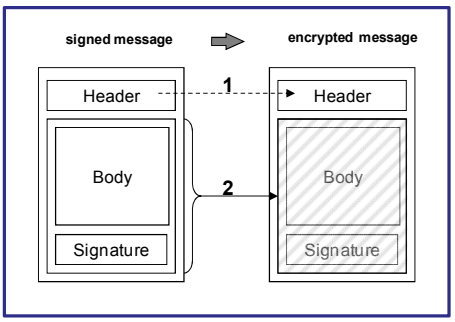


Figure 61 : Processus de chiffrement S/MIME avec la technologie SHF

Une mise en œuvre de la technologie SHF a été réalisé dans le client Trustedbird. Ces travaux ont fait l'objet d'une publication [157], fournie en annexe II. Actuellement, une soumission[54] auprès de l'IETF, est en cours. Un accord de publication du document avec un statut expérimental a été transmis par l'IETF.

8.7 *Découverte et récupération des politiques de correspondance*

Les agents en charge de l'application des politiques de correspondance doivent disposer de celles-ci. Dans le cas contraire, des mécanismes de découverte et de récupération des politiques sont nécessaires. Afin d'être diffusables, les politiques doivent, dans un premier temps, être publiées au sein d'une zone de stockage. Ensuite, il faut mettre en œuvre un mécanisme qui

[54] https://datatracker.ietf.org/doc/draft-cailleux-secure-headers/

permet la découverte des politiques de correspondance pour un domaine donné. Ceci peut être réalisé à partir de services plus ou moins propriétaires. Le service de nom du domaine (au sens DNS) offre un mécanisme d'enregistrement de service et propose une solution de découverte de service décrits dans le RFC 2782 [160]. Nous nous baserons dessus pour récupérer le nom du serveur en charge du stockage des politiques de correspondance.

L'enregistrement de service, également appelé enregistrement SRV, est un type de donnée du DNS qui permet d'indiquer la localisation d'un service. L'objectif est de définir un enregistrement SRV pour le service de diffusion de politiques de correspondance. Celui-ci pourrait prendre la forme suivante :

_correspondence._tcp.example.com. 86400 IN SRV 0 33 9020 server1.example.com.

Cet exemple d'enregistrement est composé de :

- *_correspondence* : nom du service
- *_tcp* : nom du protocole de transport
- *example.com.* : nom du domaine
- *86400* : TTL (Time To Live)
- *IN* : classe, à savoir Internet
- *SRV* : type d'enregistrement DNS
- *0* : priorité du serveur cible. Plus elle est faible, plus le serveur pourra être sollicité
- 33 : poids relatif pour les enregistrements de même priorité
- 9020 : port TCP sur lequel le service est disponible
- *server1.example.com.* : nom du serveur cible qui fournit le service

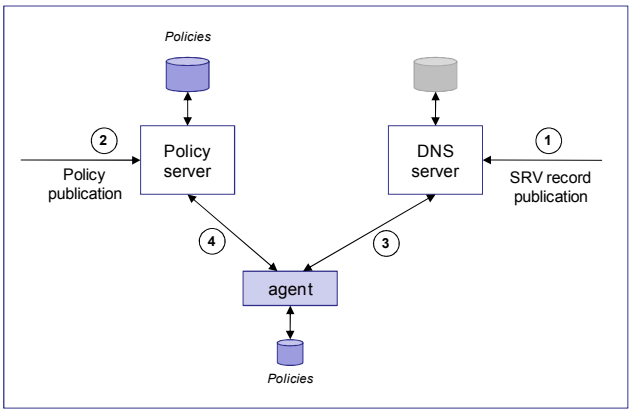


Figure 62 : Mécanismes de découverte et de récupération de politiques de correspondance

La Figure 62 décrit le scénario de découverte et de récupération des politiques de correspondance. L'enregistrement SRV est publié dans le système DNS par une personne autorisée (1). La politique est publiée dans le serveur de politiques de correspondance par une personne autorisée (2). L'agent du système de messagerie envoie une requête de découverte de service à destination du serveur DNS (3). Le serveur DNS renvoie le nom du serveur de politiques de correspondance. L'agent contacte le serveur pour récupérer la politique (4). Il

pourrait récupérer une politique particulière ou bien l'ensemble des politiques. Il est important de noter qu'il n'y a pas de mécanismes permettant de sécuriser ces échanges (contrôle d'accès, sécurisation des politiques...) définis à ce jour. Ces aspects devraient faire l'objet de futurs travaux.

8.7.1 **Récupération des capacités de correspondance**

La récupération des politiques peut être effectuée à partir d'une négociation de capacités de correspondance avec le MSA. Pour cela, l'extension CORRESPONDENCE, décrite dans le chapitre 8.4, est utilisée dans un cadre différent de celui de l'envoi d'un message. Grâce à cette extension, un utilisateur peut récupérer à partir du MSA, les capacités de correspondance dont il dispose. Le MUA peut ensuite procéder à une installation automatique des capacités.

Le principe de fonctionnement en est le suivant :

- le MUA ouvre une connexion avec le MSA,
- l'utilisateur s'authentifie auprès du MSA,
- le MUA envoie la commande CORRESPONDENCE suivi du paramètre CAPABILITY,
- le MSA transmet les correspondances autorisées pour cet utilisateur,
- le MUA et le MSA ferment la session.

La réponse à la commande CORRESPONDENCE CAPABILITY est composée de l'ensemble des correspondances autorisées. Chaque capacité est constituée du mot clé CORRESPONDENCE suivi du nom de la correspondance, de la version, du nom de domaine et d'une URL[55] (Uniform Resource Locator). Les trois premières informations permettent d'identifier la correspondance électronique. Elles sont obligatoires. La dernière information est optionnelle. Elle permet au MUA de disposer d'une adresse permettent de télécharger l'instance de la correspondance. Cette récupération de capacités est à l'initiative du MUA. Elle peut s'effectuer au lancement du MUA mais également en fonction d'un délai défini. Ce principe de fonctionnement est une alternative à la découverte de politique proposé par le service DNS. Cependant, il est limité au MUA car accessible seulement à partir du protocole SUBMISSION. Ce principe peut être intéressant pour une organisation qui ne peut pas publier dans le DNS.

Afin de garantir la confidentialité des capacités de correspondance transmises par le MSA, la transaction peut être précédée d'une négociation TLS (Transport Layer Security) [10] basée sur l'extension STARTTLS [84]. Un fonctionnement alternatif permettrait de ne présenter que certaines capacités de correspondance avant la mise en œuvre du mécanisme STARTTLS puis de présenter un ensemble plus large de capacités de correspondance après l'utilisation du mécanisme STARTTLS.

La description détaillée ci-dessous correspond à l'exemple proposé dans la Figure 63.

1. Le serveur est en attente d'une ouverture de connexion
2. Le MUA ouvre une connexion avec le MSA sur le port 587
3. Le MSA répond à la demande d'ouverture de session

[55] Le sigle URL désigne une chaîne de caractères utilisée pour adresser les ressources du World Wide Web (wikipedia).

4. le MUA envoie une commande EHLO pour débuter la transaction SUBMISSION
5. Le MSA transmet les extensions qu'il supporte au MUA
6. L'utilisateur s'authentifie auprès du MSA
7. Le MSA valide l'authentification
8. Le MUA émet une requête de capacités de correspondance
9. Le MSA transmet les capacités de correspondance au MUA
10. Le MUA envoie une fin de session
11. le MSA ferme la connexion

```
1   : <waits for connection on TCP port 587>
2   : <opens connection>
3   : 220 foo.com Simple Mail Transfer Service Ready
4   : EHLO bar.com
5   : 250-foo.com greets bar.com
5   : 250-AUTH GSSAPI DIGEST-MD5 PLAIN
5   : 250-ENHANCEDSTATUSCODES
5   : 250-STARTTLS
5   : 250-CORRESPONDENCE
5   : 250 HELP
6   : AUTH PLAIN dGVzdAB0ZXN0ADEyMzQ=
7   : 235 2.7.0 Authentication successful
8   : CORRESPONDENCE CAPABILITY
9   : 250-CORRESPONDENCE NAME=MINDAC VERSION=2.1
9   : 250-CORRESPONDENCE NAME=OFFICIAL VERSION=1.0
9   : 250-CORRESPONDENCE NAME=OPS-GEO VERSION=0.1 URL=http://corserver/geo
9   : 250 OK
10  : QUIT
11  : 221 smtp.example.com Service closing transmission channel
```

Figure 63 : Transaction SMTP intégrant une récupération des capacités de correspondance

Le MUA dispose ensuite de suffisamment d'information pour récupérer, installer puis utiliser ces types de correspondance.

8.8 *Développement d'un prototype de client de messagerie*

Un prototype de client de messagerie a été développé. Ce client, appelé *TrustedBird*[56] [161][162], est basé sur le logiciel *Mozilla Thunderbird*[57]. le client *Trustedbird* a été développé dans le cadre d'un projet piloté par la DGA (Direction Générale de l'Armement). L'objectif initial de ce client était de fournir des services de sécurité étendus[77] (reçu signé, libellé de sécurité, triple enveloppe S/MIME, SASL External...). Il embarque sous la forme d'extension, deux modules offrant des mécanismes de traitements des correspondances ; un CEA et un CR.

Les fonctionnalités offertes par *Trustedbird* sont les suivantes :

- affichage de formulaires de création et de visualisation de message basés sur des modèles

[56] http://adullact.net/plugins/mediawiki/wiki/milimail/index.php/Trustedbird_Project/fr

[57] Le client Mozilla Thunderbird est un client de messagerie libre distribué gratuitement par la Fondation Mozilla et issu du Projet Mozilla. https://www.mozilla.org/fr/thunderbird/

- génération de messages de type XIMF
- sécurisation de messages de type XIMF
- négociation de la correspondance à partir de l'extension du standard SUBMISSION
- récupération des capacités de correspondance

La création ou la visualisation de messages de type XIMF impose l'utilisation de formulaires particuliers. Trustedbird propose cette fonctionnalité. Ces formulaires sont décrits en XML. Une première version de CEA a été développé pour *Trustedbird*. Le CEA accède à des conteneurs qui embarquent la description des formulaires de création et de visualisation ainsi que les politiques simples de correspondance pour ces deux évènements. L'utilisateur peut visualiser les conteneurs installés dans le client de messagerie. Pour cela, il doit accéder aux paramètres de configuration du compte utilisateur (cf. Figure 64).

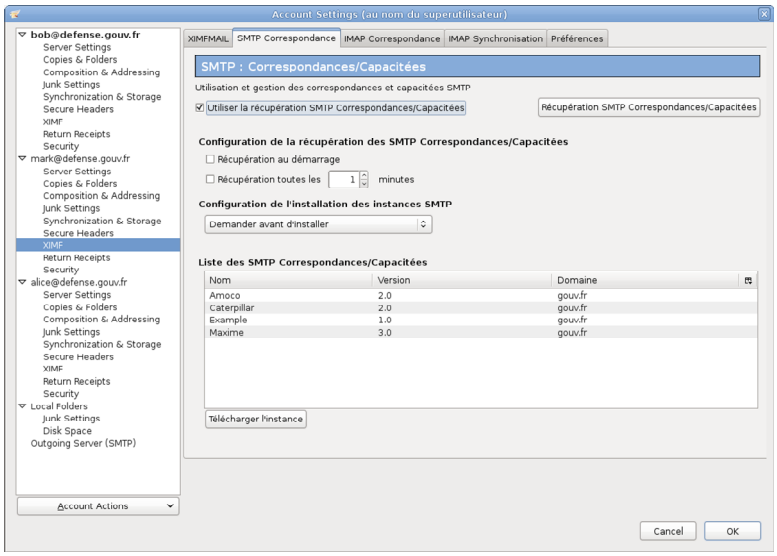


Figure 64 : fenêtre de visualisation des paramètres XIMF

Pour utiliser la fonctionnalité de formulaires spécifiques, l'utilisateur peut sélectionner un formulaire à partir du menu Ecrire. Une liste des formulaires présents est proposée. Il sélectionne ensuite le formulaire qu'il souhaite utilisé. A l'issue de cette action, un formulaire étendu s'affiche. La Figure 65 propose un exemple de formulaire étendu permettant de créer des messages militaires. Nous pouvons remarquer l'ajout d'un bandeau d'extension du formulaire. Ce bandeau est constitué d'onglets, ce qui offre des possibilité d'extension relativement importantes. Ensuite, l'utilisateur renseigne le formulaire puis sélectionne l'action Envoyer. Si tous les champs obligatoires ont été renseignés, le message est créé. Dans le cas où certains champs obligatoires n'ont pas été renseignés, le logiciel affiche une fenêtre d'avertissement. La création du message consiste à appliquer les mécanismes de sécurisation en fonction de la politique.

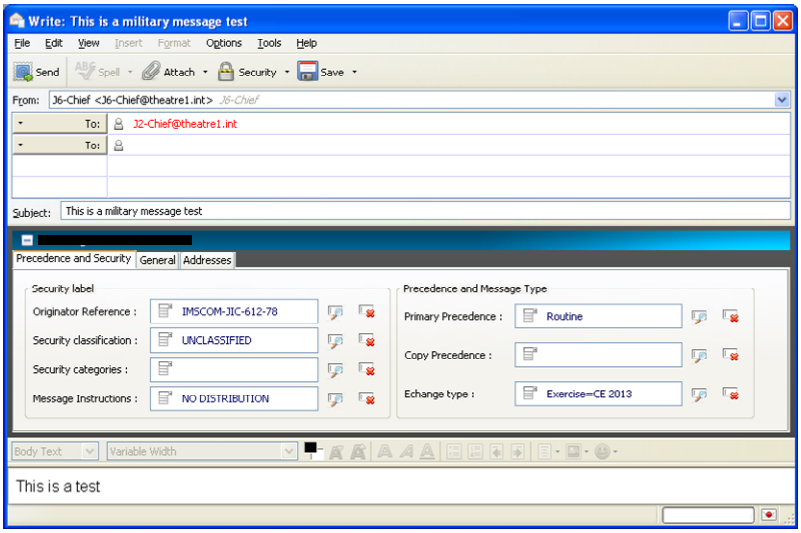


Figure 65 : formulaire de création de message

Cette action d'envoi de message va permettre :

- de générer un message de type XIMF
- d'ouvrir une connexion avec le MSA
- de négocier de la correspondance
- de soumettre le message

Ensuite le message est acheminé vers son destinataire. Si le MUA du destinataire dispose des mécanismes de détection de la correspondance alors il charge le formulaire adéquat à partir d'une base locale et affiche le message XIMF.

En complément de l'affichage des données XIMF, *Trustedbird* applique des mécanismes de contrôle de la politique. Il peut notamment vérifier une signature cryptographique et de s'assurer de la présence des champs d'entête sécurisés. Pour visualiser en détail les informations de sécurité, l'utilisateur peut ouvrir une fenêtre d'informations (fenêtre gauche de la Figure 66). Il peut ainsi visualiser différentes informations de sécurité comme l'auteur de la signature, son certificat numérique, la présence de champs d'entête sécurisés. A partir de cette même fenêtre d'informations, l'utilisateur peut découvrir l'ensemble des champs d'entête sécurisés, les statuts de chaque champ et l'algorithme de canonisation. La fenêtre droite de la Figure 66 un exemple d'affichage.

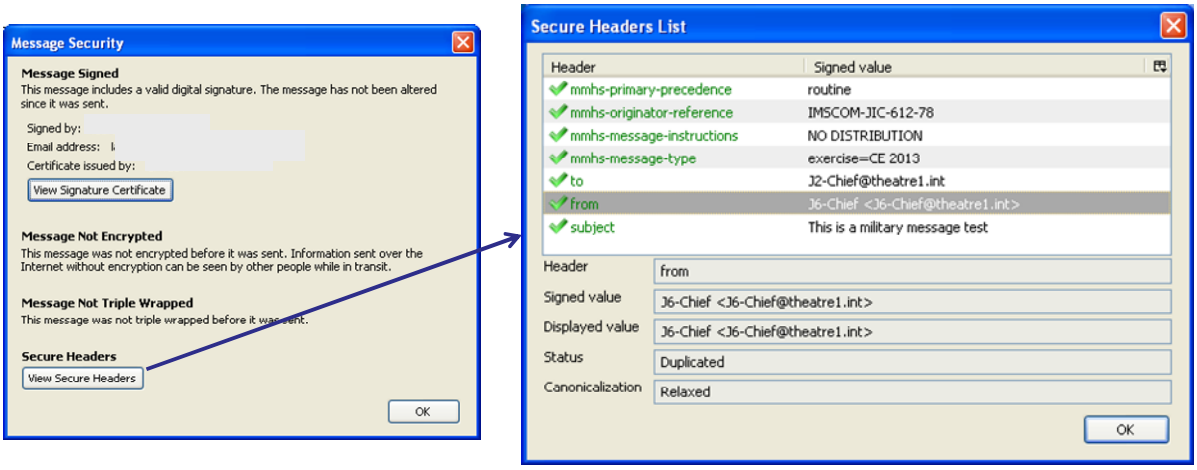


Figure 66 : Fenêtre d'information des champs d'entête sécurisés

Le client de messagerie Trustedbird développé dans le cadre de cette thèse est encore dans une phase de développement. L'intégration d'un CEA embarquant les API nécessaires à l'application des politiques de correspondances est en cours d'étude. D'autres services seront également prochainement intégrés dans le client.

8.9 *Bilan*

Dans ce chapitre, nous avons présenté une architecture basée sur le modèle de correspondance électronique. Nous avons décrit en détail les services et extensions nécessaires à l'application des concepts de correspondance électronique dans une architecture basée sur les standards. Trois extensions ont été définies :

- Une extension au standard SUBMISSION a été décrite. Elle permet de négocier la correspondance électronique courante lors de la soumission du message et d'appliquer les politiques de correspondance appropriées. Cette extension offre également un mécanisme de récupération des capacités de correspondance pour un utilisateur donné.

- Une extension au standard IMF a été décrite. Elle permet d'identifier la correspondance courante. Le principe proposé est l'ajout, lors de l'évènement de création, des informations dans un champ dédié de l'entête du message. Ceci garantit sa transmission jusqu'au destinataire final. La présence de ce champ dans le message permet aux différents agents du système de messagerie de détecter la référence de correspondance et d'appliquer la politique appropriée. Un message comportant le champ d'identification de la correspondance courante est appelé message de type XIMF. Le message peut comporter d'autres métadonnées dans l'entête du message ou de parties spécifiques dans le corps de ce même message conformément aux concepts de messages semi-structurés.

- Une extension au standard S/MIME qui nativement ne garantit pas la sécurisation des champs d'entête présents dans le message, a été introduite. Elle propose un mécanisme permettant de sécuriser la totalité des données présentes dans le message étendu. L'extension décrite dans cette thèse est appelée SHF (*Securing Header Fields*).

Les travaux de définition d'architecture ont également abouti à la description d'un mécanisme de découverte et de récupération des politiques de correspondance.

Enfin, un prototype de client de messagerie intégrant les extensions proposées a été développé. Ce client appelé *Trustedbird* est basé sur le logiciel *Mozilla Thunderbird*. Le client *Trustedbird* embarque, sous la forme d'extension, les modules permettant de traiter les correspondances de type RGS et MMHS.

Chapitre 9

Conclusion et perspectives

9.1 ***Synthèse***

Le service de messagerie électronique a contribué au formidable développement de l'Internet. Les principales raisons sont l'utilisation de protocoles de communication standardisés et ouverts, une architecture décentralisée, la simplicité, la souplesse et l'efficacité communicationnelle. Ce succès a été accompagné d'une intégration dans des solutions du type messageries collaboratives. Cependant et comme cela a été décrit dans ce rapport, ce service fait face à des limitations importantes. Il n'y a pas, par exemple, de séparation claire entre les échanges privés et professionnels, ce qui ne permet pas de différencier les usages. Les systèmes de messagerie professionnelle remédient souvent à ces déficiences en introduisant des fonctionnalités sophistiquées dans une seule politique monolithique qui devrait prendre en compte les exigences de sécurité répondant aux différents besoins. Le principal inconvénient de cette approche est lié à sa rigidité inhérente.

Les usages de la messagerie peuvent donc prendre différentes formes. Les échanges peuvent revêtir un caractère privé, cas des échanges de messages entre amis mais aussi des échanges avec son médecin ou une administration. Les échanges peuvent également concerner des aspects professionnels. D'ailleurs, la communication par messagerie électronique tient une place importante dans l'ensemble de la communication en entreprise. Ces échanges professionnels n'imposent pas les mêmes contraintes et exigences. De plus, la présence de nouvelles menaces, entrainant des attaques toujours plus sophistiquées, impose des mesures adaptées à ces différents types d'environnement.

Dans les systèmes actuels, il n'y a pas de prise en compte des concepts d'usage et de contexte qui sont pourtant fondamentaux dans les services modernes. Afin de répondre à ces limitations, un nouveau concept est proposé dans ce rapport : le concept de correspondance applicable à un système de messagerie électronique. Le modèle présenté dans ce rapport permet d'étendre les principes d'architectures de messagerie de l'Internet [6] proposé par l'IETF.

Les systèmes actuels proposent en général un modèle de correspondance unique. Il n'y a pas de prise en compte de l'intention de communication de l'utilisateur (privée ou professionnelle). La politique appliquée sur le message et lors de son acheminement, sera en général la même. Certes, l'utilisateur peut, à partir de son logiciel de messagerie, sélectionner des options particulières, ce qui entrainera des traitements spécifiques lors de la génération du message. Mais ce principe n'offrira pas obligatoirement un haut niveau d'assurance. Prenons par exemple le cas de la signature cryptographique des messages. Son utilisation simple ne propose pas obligatoirement un haut niveau d'assurance. Les critères comme les algorithmes utilisés, les extensions de sécurité, le modèle d'infrastructure de gestion des clés, sont autant de points que le système pourrait prendre en compte.

L'objectif de cette thèse était de définir un modèle de correspondance applicable à un système de messagerie électronique. Les premiers travaux ont tout d'abord consisté à analyser les concepts de la messagerie sécurisée et son contrôle par des politiques. Ils ont ensuite permis d'étudier l'application des principes de la théorie de la communication interpersonnelle dans un système de messagerie électronique. Ces travaux ont permis d'identifier le concept de correspondance dans les systèmes de messagerie.

Nous avons défini une **correspondance comme un modèle abstrait qui permet de définir et d'appliquer des politiques appropriées sur le message et lors de sa transmission pour chaque usage et en fonction du contexte courant de l'émetteur**.

Une correspondance est caractérisée par l'émission d'un message créé par un auteur, à destination d'un ou plusieurs destinataires. Cette transmission est rendue possible à partir de l'exécution d'un ensemble ordonné d'évènements qui constitue le cycle de vie d'un message.

La suite des travaux a consisté à la définition du concept de politique de correspondance. La politique de correspondance permet d'exprimer des règles et des contraintes qui seront appliquées sur le message et lors de sa transmission. Pour cela, une politique de correspondance est composée d'un ensemble de politiques évènementielles, qui seront appliquées dans chaque évènement du cycle de vie. Chaque politique évènementielle est dédiée à un évènement du cycle de vie du message. Le modèle de correspondance proposé dans ce rapport permet d'appliquer des politiques en fonction du contexte courant et des usages que les utilisateurs font du système de messagerie.

La mise en œuvre du modèle de correspondance a imposé l'extension de standards existants. La première extension permet de négocier la correspondance électronique entre un client et un serveur. Cette extension étend le standard de soumission des messages SUBMISSION [7]. La deuxième extension propose l'ajout des informations de correspondance électronique dans un message. Ce principe est rendu possible grâce à l'extension du standard IMF [9]. La troisième extension permet de sécuriser les champs d'entête présents dans les messages conformes au standard IMF. Un processus de soumission de cette extension du standard S/MIME [11] a été initié auprès de l'IETF. Un accord de publication du document avec un statut expérimental a été transmis par l'IETF.

Un principe d'architecture de système de messagerie basé sur le modèle de correspondance a également été proposé dans ce rapport. Grâce à ce principe, il est possible d'ajouter à un système conforme aux standards et sans remettre en cause son fonctionnement nominal, des mécanismes particuliers prenant en compte différents usages et contextes.

Deux prototypes ont été développés dans le cadre de cette thèse :

- Un premier développement a concerné un client de messagerie. Ce client, nommé *Trustedbird*[58], intègre un composant qui applique des politiques simples de correspondance et un composant qui embarque les politiques. Ce client de messagerie

[58] Trustedbird est un projet piloté par la DGA (Direction Générale de l'Armement). Ce logiciel, basé sur le client Mozilla Thunderbird (client de messagerie libre distribué gratuitement par la Fondation Mozilla et issu du Projet Mozilla), met en œuvre des services spécifiques ainsi que des services de sécurité avancés. Le projet est accessible à partir de l'adresse https://www.trustedbird.org

dispose également de l'extension CORRESPONDENCE et des capacités de format de messages étendus.

- Le second développement, réalisé dans le cadre d'un projet nommé AUDIENCE[59], a concerné un serveur de soumission. Ce serveur intègre l'extension CORRESPONDENCE et permet également de s'interfacer avec une architecture d'autorisation.

9.2 *Perspectives*

Les travaux ont consisté à définir un modèle de correspondance applicable à un système de messagerie électronique. Ces travaux ont ouvert un vaste chantier et laisse présager de futures perspectives intéressantes, à la fois pour le domaine commercial mais également pour le domaine de la recherche.

Un fournisseur de service de messagerie pourra proposer des offres adaptées aux besoins de ses clients. Il pourra par exemple, mettre à disposition des offres professionnelles qui intégreront des services avancés, lesquelles cohabiteront avec des offres orientées vers les usages privés. Le tout garantissant un respect du cadre juridique et légal en vigueur.

Le domaine militaire pourrait également se baser sur les concepts proposés dans cette thèse afin de définir des services de messagerie répondant à des besoins opérationnels. Il serait ainsi possible de proposer des offres en adéquation avec les spécifications définies par l'OTAN.

Les concepts proposés dans cette thèse peuvent faire l'objet d'améliorations et de futurs travaux de recherche.

- Il serait intéressant d'initier auprès de l'IETF, un processus de standardisation des extensions proposées dans cette thèse, à savoir l'extension du standard SUBMISSION et l'extension du standard IMF.
- Le principe d'architecture impose l'application de la politique de correspondance dans des composants liés aux agents du système de messagerie. Une délégation d'application de politiques permettrait de déployer ce type de solution dans des terminaux mobiles ne pouvant pas intégrer toutes les API nécessaires au contrôle de la politique.
- Les aspects sécurité sont primordiaux et pourraient également faire l'objet de futurs travaux. La sécurisation des politiques, de leurs diffusions et de leurs applications représentent des pistes à étudier.
- Le modèle de correspondance exposé dans ce rapport est limité à un périmètre intra domaine. Il serait intéressant de l'étendre à un environnement inter domaine et d'étudier les mécanismes de négociation de politiques dans un tel environnement. Cela permettrait notamment une distribution automatique des politiques entre domaines différents.
- L'expression des politiques ou les actions présentes dans les politiques pourraient évoluer. Ceci risque d'entrainer des problèmes de compatibilité entre les politiques et le

[59] AUDIENCE (Architecture de soUmission DIfferENciee de Courrier Electronique) est un projet de l'Institut Mines-Telecom.

composant en charge de l'application de celles-ci. Ce point pourrait être étudié en détail afin de garantir une meilleure comptabilité.

Les concepts proposés dans cette thèse ouvrent également d'autres perspectives concernant la définition de système de messagerie de confiance, domaine qui a fait l'objet de nombreux travaux. Deux approches principales pour gérer la confiance ont été identifiées [163]; une approche basée sur la réputation et une approche basée sur la politique. En se basant sur la seconde, il serait possible de définir l'ensemble des services répondant à des usages particuliers et de décrire les politiques à appliquer pour chaque usage. Ce principe ouvre la voie à de nouvelles solutions de messagerie de confiance.

Parallèlement, il est important de souligner que la langue française n'utilise qu'un mot pour désigner la confiance alors que les anglo-saxons en utilisent deux : *trust* et *confidence*. N. Luhmann a proposé la distinction suivante entre ces deux termes [164].

The distinction between confidence and trust depends on our ability to distinguish between dangers and risks, whether remote or a matter of immediate concern.

Trust renvoie à un sentiment de confiance qui inclut une prise de risque. Il ne peut y avoir de certitude. Alors que le terme *confidence* est à rapprocher de la notion d'assurance. Cette distinction, permet de traduire *trust* par "confiance" et *confidence* par "assurance". Le concept de *trust* implique une prise de risque alors que le concept *confidence* pourrait être associé à une idée d'assurance. L'application de ces concepts dans les systèmes de messagerie trouve tout son intérêt. Nous pourrions imaginer un système fournissant différents services visant à minimiser la prise de risque en sélectionnant la bonne politique pour un usage et un contexte donnés. C'est l'exacte et systématique adéquation entre le besoin de l'utilisateur et la politique fournie qui assure une prise de risque limitée et donc le caractère d'assurance du service de messagerie. D'un autre côté, la qualification de confiance viendrait de la prise de risque liée à l'application d'une politique monolithique statique pas toujours adaptée à chaque usage et au contexte.

Les concepts de correspondance électronique proposés dans cette thèse ouvrent de nouvelles perspectives et permettent d'envisager la définition de futurs systèmes de messagerie avancée.

BIBLIOGRAPHIE

[1] Miles Tracy, Wayne Jansen, Karen Scarfone, and Jason Butterfield, Guidelines on Electronic Mail Security, February 2007, Special Publication 800-45 Version 2.

[2] NATO, Requirements and Rationale for Future Military Messaging, February 2005.

[3] P. Hartley, *Interpersonal Communication*.: Routledge, 1999.

[4] J. Searle, *Speech Acts: an essay in the philosophy of language*. Cambridge: Cambridge University Press, 1969.

[5] ANSSI, Référentiel Générale de Sécurité, May 2010, v1.0.

[6] D. Crocker, Internet Mail Architecture, july 2009, RFC 5598 (Informational).

[7] R. Gellens and J. Klensin, Message Submission for Mail, november 2011, RFC 6409 (Standard).

[8] J. Klensin, Simple Mail Transfer Protocol, october 2008, RFC 5321 (Draft Standard).

[9] P. Resnick, Internet Message Format, october 2008, RFC 5322 (Draft Standard).

[10] T. Dierks and E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.2, august 2008, RFC 5246 (Proposed Standard), Updated by RFCs 5746, 5878, 6176.

[11] B. Ramsdell and S. Turner, Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification, january 2010, RFC 5751 (Proposed Standard).

[12] R.W. Watson, Mail Box Protocol, july 1971, RFC 196, Obsoleted by RFC 221.

[13] Craig Partridge, "The technical development of internet email," *IEEE Annals of the History of Computing*, vol. 1, no. 2, pp. 3-29, 2008.

[14] N. Freed and N. Borenstein, Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, november 1996, RFC 2045 (Draft Standard), Updated by RFCs 2184, 2231, 5335.

[15] N. Freed and N. Borenstein, Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types, november 1996, RFC 2046 (Draft Standard), Updated by RFCs 2646, 3798, 5147.

[16] K. Moore, MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text, november 1996, RFC 2047 (Draft Standard), Updated by RFCs 2184, 2231.

[17] N. Freed, J. Klensin, and J. Postel, Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures, november 1996, RFC 2048 (Best Current Practice), Obsoleted by RFCs 4288, 4289, updated by RFC 3023.

[18] N. Freed and N. Borenstein, Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples, november 1996, RFC 2049 (Draft Standard).

[19] S. Farrell, R. Housley, and S. Turner, Simple Mail Transfer Protocol Extension for Message Transfer Priorities, august 2010, RFC 6710 (Proposed Standard).

[20] J. Klensin, Application Techniques for Checking and Transformation of Names, february 2004, RFC 3696 (Informational).

[21] A.K. Bhushan, K.T. Pogran, R.S. Tomlinson, and J.E. White, Standardizing Network Mail Headers, september 1973, RFC 561.

[22] D. Crocker, STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES, august 1982, RFC 822 (Standard).

[23] P. Resnick, Internet Message Format, april 2001, RFC 2822 (Proposed Standard).

[24] J. Postel, Simple Mail Transfer Protocol, august 1982, RFC 821 (Standard), Obsoleted by RFC 2821.

[25] P. Resnick, Internet Message Format, april 2001, RFC 2822 (Proposed Standard).

[26] M Tariq Banday, Jameel A Qadri, and Nisar A Shah, "A Practical Study of E-mail Communication through SMTP," 2010.

[27] J. Myers and M. Rose, Post Office Protocol - Version 3, may 1996, RFC 1939 (Standard).

[28] M. Crispin, INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1, march 2003, RFC 3501 (Proposed Standard), Updated by RFCs 4466, 4469, 4551, 5032, 5182, 5738, 6186.

[29] C. Hutzler, D. Crocker, P. Resnick, E. Allman, and T. Finch, Email Submission Operations: Access and Accountability Requirements, november 2007, RFC 5068 (Best Current Practice).

[30] P.V. Mockapetris, Domain names - implementation and specification, november 1987, RFC 1035 (Standard).

[31] K. Moore and G. Vaudreuil, An Extensible Message Format for Delivery Status Notifications, january 2003, RFC 3464 (Draft Standard), Updated by RFCs 4865, 5337.

[32] K. Moore, Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notifications (DSNs), january 2003, RFC 3461 (Draft Standard), updated by RFCs 3798, 3885, 5337.

[33] T. Hansen and G. Vaudreuil, Message Disposition Notification, may 2004, RFC 3798 (Draft Standard).

[34] E. Allman and T. Hansen, SMTP Service Extension for Message Tracking, september 2004, RFC 3885 (Proposed Standard).

[35] John AA Sillince, Stuart Macdonald, Bernard Lefang, and Brian Frost, "Email adoption, diffusion, use and impact within small firms: A survey of UK companies," *International Journal of Information Management*, vol. 18, no. 4, pp. 231-242, 1998.

[36] Laura A Dabbish and Robert E Kraut, "Email overload at work: an analysis of factors associated with email strain," in *Proceedings of the 2006 20th anniversary conference on Computer*

supported cooperative work, 2006, pp. 431-440.

[37] F.B. Schneider and National Research Council (U.S.). Committee on Information Systems Trustworthiness, *Trust in cyberspace.*: National Academy Press, 1999.

[38] Mouna Jouini, Latifa Ben Arfa Rabai, and Anis Ben Aissa, "Classification of Security Threats in Information Systems," *Procedia Computer Science*, vol. 32, pp. 489-496, 2014.

[39] M. Rogers, *Internal security threats.*: John Wiley and Sons, 2006, vol. Volume III: Threats, Vulnerabilities,Prevention, Detection, and Management.

[40] Ali Abbas, Abdulmotaleb El-Saddik, and Ali Miri, "A Comprehensive Approach to Designing Internet Security Taxonomy.," in *CCECE*, 2006, pp. 1316-1319.

[41] D. Harley, *E-Mail Threats and Vulnerabilities.*: John Wiley and Sons, 2006, vol. Volume III: Threats, Vulnerabilities,Prevention, Detection, and Management.

[42] SPAMHAUS, The definition of spam, http://www.spamhaus.org/definition.html.

[43] Stephen Hinde, "Spam: the evolution of a nuisance," *Computers and Security*, vol. 22, no. 6, pp. 474-478, 2003.

[44] Jason Hong, "The State of Phishing Attacks," *Commun. ACM*, vol. 55, no. 1, pp. 74-81, january 2012.

[45] R. Shirey, Internet Security Glossary, Version 2, august 2007, RFC 4949 (Informational).

[46] ISO, Information technology - Security techniques - Information security management systems - Overview and vocabulary - ISO IEC 27000, January 2014.

[47] William Stallings, *Network Security Essentials: Applications and Standards. New Jersey, Prentice-Hall Inc.*, 2000.

[48] Dan Zhou, Joncheng C. Kuo, Susan Older, and Shiu kai Chin, "Formal Development of Secure Email," in *In Proceedings of the 32nd Hawaii International Conference on System Sciences*, 1999.

[49] Simson L Garfinkel, David Margrave, Jeffrey I Schiller, Erik Nordlander, and Robert C Miller, "How to make secure email easier to use," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2005, pp. 701-710.

[50] Ed Gerck, "Secure Email Technologies X. 509/PKI, PGP, IBE and Zmail," *Corporate Email Management*, pp. 171-196, 2007.

[51] ITU, "Security architecture for Open Systems Interconnection for CCITT applications," march 1991.

[52] Sean Turner and Russ Housley, *Implementing Email and Security Tokens: Current Standards, Tools, and Practices.*: John Wiley \& Sons, 2008.

[53] Ammar Almomani, BB Gupta, Samer Atawneh, A Meulenberg, and Eman Almomani, "A survey of phishing email filtering techniques," *Communications Surveys and Tutorials, IEEE*, vol. 15, no. 4, pp. 2070-2090, 2013.

[54] Bimal Parmar, "Protecting against spear-phishing," *Computer Fraud and Security*, vol. 2012, no. 1, pp. 8-11, 2012.

[55] Gordon V Cormack, "Email spam filtering: A systematic review," *Foundations and Trends in Information Retrieval*, vol. 1, no. 4, pp. 335-455, 2007.

[56] Havard Wik Thorkildssen, "SPAM-Different Approaches to Fighting Unsolicited Commercial Email A survey of spam and spam countermeasures," *available at citeseerx. ist. psu. edu/viewdoc/summary*, 2004.

[57] Pascal Manzano, "Enisa 2009 spam survey: Measures used by providers to reduce spam," *White Paper*, 2009.

[58] Xiao-lin Wang and others, "Learning to classify email: a survey," in *2005 International Conference on Machine Learning and Cybernetics*, vol. 9, 2005, pp. 5716-5719.

[59] S. Bellovin, J. Schiller, and C. Kaufman, Security Mechanisms for the Internet, december 2003, RFC 3631 (Informational).

[60] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone, *Handbook of applied cryptography*.: CRC press, 2010.

[61] Hannane El Bakkali and Bahia Kaitouni, "L'analyse formelle du modèle de confiance d'une PKI à l'aide d'une logique de croyance," in *Actes de GRES 2001*, December 2001.

[62] D Crocker, Trust in Email Begins with Authentication, June 2008, Messaging Anti-Abuse Working Group.

[63] Bryan D Payne and W Keith Edwards, "A brief introduction to usable security," *Internet Computing, IEEE*, vol. 12, no. 3, pp. 13-21, 2008.

[64] Jason Andress, *The basics of information security: understanding the fundamentals of InfoSec in theory and practice*.: Elsevier, 2011.

[65] A. Melnikov and K. Zeilenga, Simple Authentication and Security Layer (SASL), june 2006, RFC 4422 (Proposed Standard).

[66] R. Siemborski and A. Melnikov, SMTP Service Extension for Authentication, july 2007, RFC 4954 (Proposed Standard).

[67] J. Linn, Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures, february 1993, RFC 1421 (Historic).

[68] S. Kent, Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management, february 1993, RFC 1422 (Historic).

[69] D. Balenson, Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers, february 1993, RFC 1423 (Historic).

[70] B. Kaliski, Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services, february 1993, RFC 1424 (Historic).

[71] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer, OpenPGP Message Format, november 2007, RFC 4880 (Proposed Standard), Updated by RFC 5581.

[72] Alfarez Abdul-Rahman, "The pgp trust model," in *EDI-Forum: the Journal of Electronic Commerce*, vol. 10, 1997, pp. 27-31.

[73] S. Crocker, N. Freed, J. Galvin, and S. Murphy, MIME Object Security Services, october 1995, RFC 1848 (Historic).

[74] Charles Dinkel, Secure Data Network System (SDNS) Network, Transport, and Message Security Protocols, 1990.

[75] S. Turner, "Secure/Multipurpose Internet Mail Extensions," *Internet Computing, IEEE*, vol. 14, no. 5, pp. 82-86, Sept 2010.

[76] B. Ramsdell and S. Turner, Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Certificate Handling, january 2010, RFC 5750 (Proposed Standard).

[77] P. Hoffman, Enhanced Security Services for S/MIME, june 1999, RFC 2634 (Proposed Standard).

[78] R. Housley, Cryptographic Message Syntax (CMS), september 2009, RFC 5652 (Standard).

[79] R. Housley, Cryptographic Message Syntax (CMS) Algorithms, august 2002, RFC 3370 (Proposed Standard), Updated by RFC 5754.

[80] S. Kitterman, Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1, april 2014, RFC 7208.

[81] J. Lyon and M. Wong, Sender ID: Authenticating E-Mail, april 2006, RFC 4406 (Experimental).

[82] Barry Leiba and Jim Fenton, "DomainKeys Identified Mail (DKIM): Using Digital Signatures for Domain Verification.," in *CEAS*, 2007.

[83] D. Crocker, T. Hansen, and M. Kucherawy, DomainKeys Identified Mail (DKIM) Signatures, september 2011, RFC 6376 (Draft Standard).

[84] P. Hoffman, SMTP Service Extension for Secure SMTP over Transport Layer Security, february 2002, RFC 3207 (Proposed Standard).

[85] C. Newman, Using TLS with IMAP, POP3 and ACAP, june 1999, RFC 2595 (Proposed Standard), Updated by RFC 4616.

[86] Saket Kaushik, Paul Ammann, Duminda Wijesekera, William Winsborough, and Ronald Ritchey, "A policy driven approach to email services," in *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*, 2004, pp. 169-178.

[87] Saket Kaushik, "Policy-Controlled Email Services," George Mason University, Ph.D. dissertation 2007.

[88] Gansen Zhao and David Chadwick, "Trust infrastructure for policy based messaging in open environments," in *Enabling Technologies: Infrastructure for Collaborative Enterprise, 2005. 14th IEEE International Workshops on*, 2005, pp. 144-149.

[89] The OASIS technical commitee, XACML: eXtensible Access Control Markup Language, January 2013.

[90] S. Farrell, R. Housley, and S. Turner, An Internet Attribute Certificate Profile for

Authorization, january 2010, RFC 5755 (Proposed Standard).

[91] Marco Casassa Mont, Siani Pearson, and Pete Bramhall, "Towards Accountable Management of Identity and Privacy: Sticky Policies and Enforceable Tracing Services," in *Proceedings of the 14th International Workshop on Database and Expert Systems Applications*, Washington, DC, USA, 2003, pp. 377--.

[92] Raja Afandi, Jianqing Zhang, Munawar Hafiz, and Carl A Gunter, "AMPol: Adaptive messaging policy," in *Web Services, 2006. ECOWS'06. 4th European Conference on*, 2006, pp. 53-64.

[93] Kevin Lux, Carl A. Gunter, and Michael J. May, "WSEmail: Secure internet messaging based on web services," in *In International Conference on Web Services (ICWS*, 2005, pp. 75-82.

[94] David Chadwick, Graeme Lunt, and Gansen Zhao, "Secure role based messaging," in *Communications and Multimedia Security*, 2005, pp. 263-275.

[95] Gansen Zhao, "Secure Role Based Messaging," University of Kent, Ph.D. dissertation 2009.

[96] Gansen Zhao and David W Chadwick, "Evolving messaging systems for secure role based messaging," in *Engineering of Complex Computer Systems, 2005. ICECCS 2005. Proceedings. 10th IEEE International Conference on*, 2005, pp. 216-223.

[97] David W Chadwick, Alexander Otenko, and Edward Ball, "Role-based access control with X. 509 attribute certificates," *Internet Computing, IEEE*, vol. 7, no. 2, pp. 62-69, 2003.

[98] D. Cooper et al., Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, may 2008, RFC 5280 (Proposed Standard).

[99] K. Zeilenga, Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map, june 2006, RFC 4510 (Proposed Standard).

[100] C. Chabrol and M. Radu, *Psychologie de la communication et de la persuasion: Théories et applications.*: De Boeck Supérieur, 2008.

[101] Patrick Bonin, *Psychologie du langage - la fabrique des mots*, De boeck, Ed.: Ouvertures psychologiques, 2013.

[102] J.R. Searle, *Mind, Language And Society: Philosophy In The Real World.*: Basic Books, 1999.

[103] Thierry Lemeunier, "L'intentionnalité communicative dans le dialogue homme-machine en langue naturelle," Université du maine, Ph.D. dissertation 2000.

[104] STEFAN FRYDRYCHOWICZ and JOANNA MATEJCZUK, "The role of intention in the process of interpersonal communication," *Psychology of Language and Communication*, vol. 10, no. 2, pp. 89-108, 2006.

[105] Joao Girao and Amardeo Sarma, "IDentity Engineered Architecture (IDEA).," in *Future Internet Assembly*, 2010, pp. 85-93.

[106] Olle Balter, "Electronic mail in a working context," KTH, Ph.D. dissertation 1998.

[107] Jérome Mulsant, Gaelle Recourcé, and Romain Vuillemot, "Tomorrow's E-Mail: DLM 3.0 project's vision for the future of E-Mail in enterprises," in *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-*

Volume 03, 2011, pp. 377-380.

[108] Alinto, Usage de la messagerie en entreprise, 2014.

[109] Wendy E Mackay, "Diversity in the use of electronic mail: A preliminary inquiry," *ACM Transactions on Information Systems (TOIS)*, vol. 6, no. 4, pp. 380-397, 1988.

[110] Nadia Gauducheau, "Electronic Mail Usage at Work: An Analysis of the Users' Representations and Affects," in *Proceedings of the 29th Annual European Conference on Cognitive Ergonomics*, New York, NY, USA, 2011, pp. 137-140.

[111] Catherine Grevet, David Choi, Debra Kumar, and Eric Gilbert, "Overload is Overloaded: Email in the Age of Gmail," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2014, pp. 793-802.

[112] Nicolas Ducheneaut and Victoria Bellotti, "E-mail As Habitat: An Exploration of Embedded Personal Information Management," *interactions*, vol. 8, no. 5, pp. 30-38, september 2001.

[113] Steve Whittaker, Victoria Bellotti, and Jacek Gwizdka, "Email in personal information management," *Communications of the ACM*, vol. 49, no. 1, pp. 68-73, 2006.

[114] Antonio F. Gomez-Skarmeta, Pedro Martinez-Julia, Joao Girao, and Amardeo Sarma, "Identity Based Architecture for Secure Communication in Future Internet," in *Proceedings of the 6th ACM Workshop on Digital Identity Management*, New York, NY, USA, 2010, pp. 45-48.

[115] Ordonnance n°2005-1516 du 8 décembre 2005 relative aux échanges électroniques entre les usagers et les autorités administratives et entre les autorités administratives., december 2005.

[116] C.D. Bonatti, "Medium grade features for military messaging," in *Military Communications Conference Proceedings, 1999. MILCOM 1999. IEEE*, vol. 2, 1999, pp. 1256-1261 vol.2.

[117] NATO, STANAG 4406 Edition 2, Military Message Handling System, March 2005.

[118] John Langshaw Austin, *How to do things with words.*: Oxford university press, 1975, vol. 1955.

[119] J. Searle, "A taxonomy of illocutionary acts," in *Language, Mind and Knowledge*, 1975, pp. 344-369.

[120] Terry Winograd and Fernando Flores, *Understanding computers and cognition: A new foundation for design.*: Intellect Books, 1986.

[121] Terry Winograd, "A language/action perspective on the design of cooperative work," *Human--Computer Interaction*, vol. 3, no. 1, pp. 3-30, 1987.

[122] William W Cohen, Vitor R Carvalho, and Tom M Mitchell, "Learning to Classify Email into``Speech Acts".," in *EMNLP*, 2004, pp. 309-316.

[123] Paul N Bennett and Jaime Carbonell, "Detecting action-items in e-mail," in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, 2005, pp. 585-586.

[124] Vitor R. Carvalho, "Modeling Intention in Email," Carnegie Mellon University, Ph.D. dissertation 2008.

[125] Luke Mcdowell, Oren Etzioni, Alon Halevy, and Henry Levy, "Semantic email," in *WWW '04: Proceedings of the 13th international conference on World Wide Web*, New York, NY, USA, 2004, pp. 244-254.

[126] Luke McDowell, Oren Etzioni, and Alon Halevy, "Semantic Email: Theory and Applications," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 2, no. 2, 2004.

[127] Oren Etzioni, Alon Y Halevy, Henry M Levy, and Luke McDowell, "Semantic Email: Adding Lightweight Data Manipulation Capabilities to the Email Habitat.," in *WebDB*, 2003, pp. 13-18.

[128] Tim Berners-Lee, James Hendler, Ora Lassila, and others, "The semantic web," *Scientific american*, vol. 284, no. 5, pp. 28-37, 2001.

[129] Luke McDowell, Oren Etzioni, and Alon Y Halevy, "Specifying Semantic Email Processes.," in *WWW Workshop On Application Design, Development and Implementation Issues in the Semantic Web*, 2004.

[130] Simon Scerri, Brian Davis, Siegfried Handschuh, and Manfred Hauswirth, "Semanta - Semantic Email Made Easy.," in *ESWC*, vol. 5554, 2009, pp. 36-50.

[131] Simon Scerri, Ioana Giurgiu, Brian Davis, and Siegfried Handschuh, "Semanta--Semantic Email in Action," in *The Semantic Web: Research and Applications*.: Springer, 2009, pp. 883-887.

[132] T. W. Malone, K. R. Grant, and F. A. Turbak, "The Information Lens: An Intelligent System for Information Sharing in Organizations," *SIGCHI Bull.*, vol. 17, no. 4, pp. 1-8, april 1986.

[133] Thomas W Malone, Kenneth R Grant, Franklyn A Turbak, Stephen A Brobst, and Michael D Cohen, "Intelligent information-sharing systems," *Communications of the ACM*, vol. 30, no. 5, pp. 390-402, 1987.

[134] Thomas W. Malone, Kenneth R. Grant, Kum-Yew Lai, Ramana Rao, and David Rosenblitt, "Semi-structured Messages Are Surprisingly Useful for Computer-supported Coordination," in *Proceedings of the 1986 ACM Conference on Computer-supported Cooperative Work*, New York, NY, USA, 1986, pp. 102-114.

[135] Vitor R Carvalho and William W Cohen, "Preventing Information Leaks in Email.," in *SDM*, 2007, pp. 68-77.

[136] S Baskaran, "Content based email classification system by applying conceptual maps," in *Intelligent Agent and Multi-Agent Systems, 2009. IAMA 2009. International Conference on*, 2009, pp. 1-2.

[137] Jason Pascoe, Nick Ryan, and David Morse, "Issues in developing context-aware computing," in *Handheld and ubiquitous computing*, 1999, pp. 208-221.

[138] Bernard Kerr and Eric Wilcox, "Designing remail: reinventing the email client through innovation and integration," in *CHI'04 Extended Abstracts on Human Factors in Computing Systems*, 2004, pp. 837-852.

[139] Daniel Gruen et al., "Lessons from the reMail prototypes," in *Proceedings of the 2004 ACM*

conference on Computer supported cooperative work, 2004, pp. 152-161.

[140] Steven L Rohall, Daniel Gruen, Paul Moody, and Seymour Kellerman, "Email visualizations to aid communications," in *Proceedings of InfoVis 2001 The IEEE Symposium on Information Visualization, IEEE*, vol. 12, 2001, p. 15.

[141] Steven L Rohall, "Redesigning Email for the 21 st Century Workshop Position Paper," in *CSCW 2002 Workshop: Redesigning Email for the 21st Century. New Orleans, LA*, 2002.

[142] Anind K. Dey, "Understanding and Using Context," *Personal Ubiquitous Comput.*, vol. 5, no. 1, pp. 4-7, january 2001.

[143] Bill N Schilit and Marvin M Theimer, "Disseminating active map information to mobile hosts," *Network, IEEE*, vol. 8, no. 5, pp. 22-32, 1994.

[144] Bill Schilit, Norman Adams, and Roy Want, "Context-aware computing applications," in *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, 1994, pp. 85-90.

[145] Gregory D Abowd et al., "Towards a better understanding of context and context-awareness," in *Handheld and ubiquitous computing*, 1999, pp. 304-307.

[146] David R Morse, Stephen Armstrong, and Anind K Dey, "The what, who, where, when, why and how of context-awareness," in *CHI'00 extended abstracts on Human factors in computing systems*, 2000, pp. 371-371.

[147] Paul Prekop and Mark Burnett, "Activities, context and ubiquitous computing," *Computer Communications*, vol. 26, no. 11, pp. 1168-1176, 2003.

[148] Thomas Hofer et al., "Context-awareness on mobile devices-the hydrogen approach," in *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, 2003, pp. 10--pp.

[149] B. Leiba, IMAP4 IDLE command, june 1997, RFC 2177 (Proposed Standard).

[150] Nancy A Lynch and Mark R Tuttle, "An introduction to input/output automata," 1988.

[151] ANSSI, Référentiel Générale de Sécurité - Annexe B1 - Mécanismes cryptographiques, january 2010, v1.20.

[152] S. Turner, Using SHA2 Algorithms with Cryptographic Message Syntax, january 2010, RFC 5754 (Proposed Standard).

[153] A. Melnikov and G. Lunt, Registration of Military Message Handling System (MMHS) Header Fields for Use in Internet Mail, january 2012, RFC 6477 (Informational).

[154] L. Cailleux, A. Bouabdallah, and S. Gombault, "Architecure de soumission différenciée," 2014.

[155] R. Yavatkar, D. Pendarakis, and R. Guerin, A Framework for Policy-based Admission Control, january 2000, RFC 2753 (Informational).

[156] G. Klyne and J. Palme, Registration of Mail and MIME Header Fields, march 2005, RFC 4021 (Proposed Standard), Updated by RFC 5322.

[157] L. Cailleux, "A new security service for future military messaging," in *Military Communications and Information Systems Conference (MCC), 2013*, Oct 2013, pp. 1-8.

[158] L. Cailleux and C. Bonatti, "Securing Header Fields with S/MIME," IETF, IETF Internet Draft -- work in progress july 2014.

[159] T. Dean and W. Ottaway, Domain Security Services using S/MIME, october 2001, RFC 3183 (Experimental).

[160] A. Gulbrandsen, P. Vixie, and L. Esibov, A DNS RR for specifying the location of services (DNS SRV), february 2000, RFC 2782 (Proposed Standard).

[161] L. Cailleux, "Trustedbird, un client de messagerie de confiance," , Nantes (France), 2009.

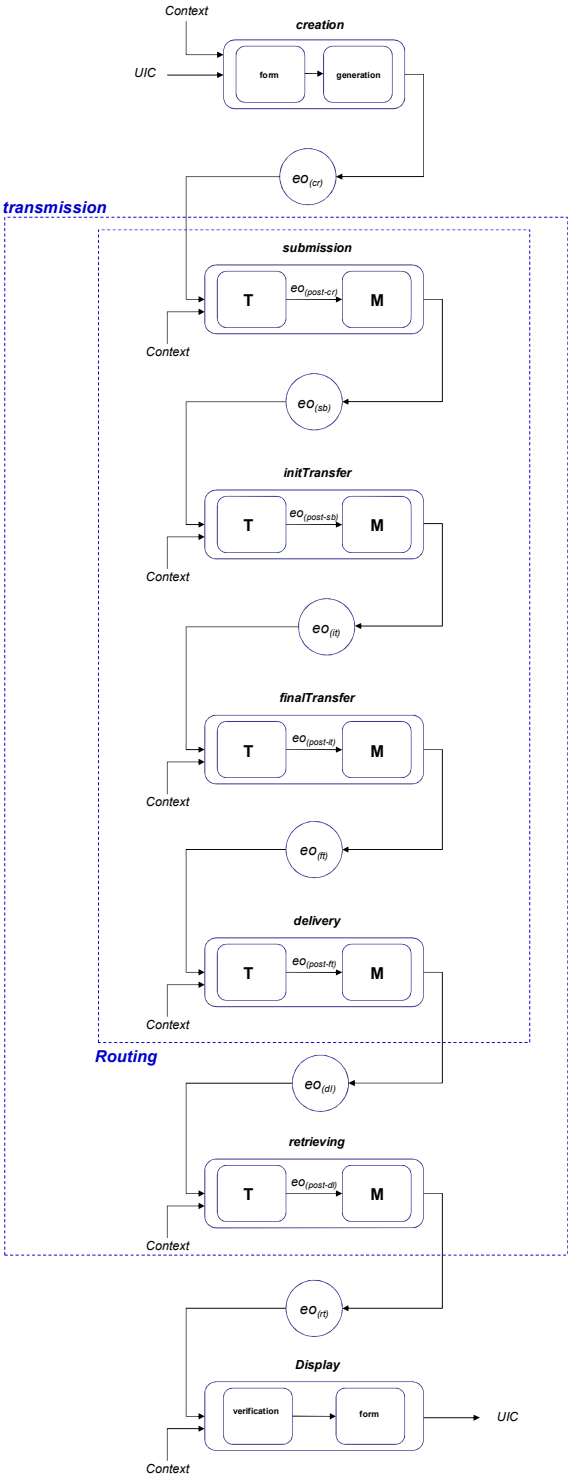
[162] L. Cailleux, Trustedbird, 2010, Trustedbird.

[163] Piero Bonatti, Claudiu Duma, Daniel Olmedilla, and Nahid Shahmehri, "An Integration of Reputation-based and Policy-based Trust Management," in *In Proceedings of the Semantic Web Policy Workshop*, 2005.

[164] Niklas Luhmann, "Familiarity, confidence, trust: Problems and alternatives," *Trust: Making and breaking cooperative relations*, vol. 6, pp. 94-107, 2000.

[165] ITU-T, X.400 - MESSAGE HANDLING AND DIRECTORY SERVICES - OPERATIONS AND DEFINITION OF SERVICE F.400, august 1992.

[166] CCEB, ACP123 (B), Common Messaging Strategy and Procedures, March 2014.


Context
creation
UIC
form
generation
eo(cr)
transmission
submission
T
eo(post-cr)
M
Context
eo(sb)
initTransfer
T
eo(post-sb)
M
Context
eo(it)
finalTransfer
T
eo(post-it)
M
Context
eo(ift)
delivery
T
eo(post-ft)
M
Context
Routing
eo(dl)
retrieving
T
eo(post-dl)
M
Context
eo(rt)
Display
verification
form
UIC
Context

ANNEXE II - SECURING HEADER FIELDS WITH S/MIME

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 26 January 2015

L. Cailleux
DGA MI
C. Bonatti
IECA
26 July 2014

Securing Header Fields with S/MIME
draft-cailleux-secure-headers-06

Abstract

This document describes how the S/MIME protocol can be extended in order to secure message header fields. This technology provides security services such as data integrity, non-repudiation and confidentiality. This extension is referred to as 'Secure Headers'.

Table of Contents

1. Introduction

S/MIME [RFC 5751] standard defines a data encapsulation format for the achievement of end to end security services such as integrity, authentication, non-repudiation and confidentiality. By default, S/MIME secures message body parts, at the exclusion of the message header fields.

S/MIME provides an alternative solution to secure header fields. "The sending client MAY wrap a full MIME [RFC 2045] message in a message/rfc822 wrapper in order to apply S/MIME security services to header fields". However, the S/MIME solution doesn't allow selection of a subset of message header fields to secure. In addition, confidentiality service can not be implemented for message header fields. The solution described herein overcomes those limitations.

Several security standards exist such as DKIM [RFC 6376], STARTTLS [RFC 3207] and TLS with IMAP [RFC 2595] but meet other needs (signing domain, secure channels). An internet draft referred to as PROTECTED HEADERS has been proposed, but doesn't address all the requirements. These different solutions are explained in the next chapters.

The goal of this document is to define end to end secure header fields mechanisms compliant with S/MIME standard. This technique is based on the signed attribute fields of a Cryptographic Message Syntax (CMS) [RFC 5652] signature.

2. Terminology and conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and

"OPTIONAL" in this document are to be interpreted as described in [RFC 2119].

MUA, MSA and MTA terms are defined in Email architecture document [RFC 5598].

DCA term is defined in the S/MIME Domain Security specification [RFC 3183].

End-to-end Internet Mail exchanges are performed between message originators and recipients.

Description of message header fields are described in [RFC 5322]. A header field is composed of a name and a value.

Secure Headers technology uses header fields statuses required to provide a confidentiality service toward message headers. The following three terms are used to describe the field statuses:

- Duplicated (the default status). When this status is present or if no status is specified, the signature process embeds the header field value in the digital signature, but the value is will also be present in the message header fields.

- Deleted. When this status is present, the signature process embeds the header field value in the digital signature, and the encryption process deletes this field from the message to preserve its confidentiality.

- Modified. When this status is present, the signature process embeds the header field value in the digital signature, and the encryption process modifies the value of the header field in the message. This preserves confidentiality and informs a receiver's non-compliant MUA that secure headers are being used. New values for each

field might be configured by the sender (i.e., "This header is secured, use a compliant client").

3. Context

Over the Internet, email usage has grown and today represents a fundamental service. Meanwhile, continually increasing threat levels are motivating the implementation of security services.

Historically, SMTP [RFC 5321] and IMF [RFC 5322] don't provide, by default, security services. The S/MIME standard [RFC 5751] was published in order to encompass these needs. S/MIME defines a data encapsulation format for the provision of end to end security services such as integrity, authentication, non-repudiation and confidentiality. By default, S/MIME secures message body parts, at the exclusion of the message header fields. In order to protect message header fields (for instance, the "Subject", "To", "From" or customized fields), several solutions exist.

S/MIME defines an encapsulation mechanism, chapter 3.1: "The sending client may wrap a full MIME message in a message/rfc822 wrapper in order to apply S/MIME security services to these header fields. It is up to the receiving client to decide how to present this inner header along with the unprotected outer header". However, some use cases are not addressed, especially in the case of message encryption. What happens when header fields are encrypted? How does the receiving client display these header fields? How can a subset of header fields be secured? S/MIME doesn't address these issues.

An alternative solution is described in [RFC 5750]. "Receiving agents MUST check that the address in the From or Sender header of a mail message matches an Internet mail address, if present, in the signer's certificate, if mail addresses are

present in the certificate". However, this solution only provides a matching mechanism between email addresses, and provides no protection to other header fields.

Other security standards (introduced below) exist such as DKIM, STARTTLS and TLS with IMAP but meet other needs (signing domain, secure channels...).

STARTTLS and TLS with IMAP provide secure channels between components of email system (MUA, MSA, MTA...) but end to end integrity cannot be guaranteed.

DKIM defines a domain-level authentication framework for email to permit verification of the source and contents of messages. It provides mechanisms to secure message header fields and message body but it doesn't guarantee non-repudiation and originator authentication. In addition, it doesn't provide confidentiality.

An internet draft referred to as Protected Headers (PRHDRS) has been proposed. Mechanisms described in this draft are the following. "A digest value is computed over the canonicalized version of some selected header fields. This technique resembles header protection in DKIM. Then the digest value is included in a signed attribute field of a CMS signature". This specification doesn't address all conceivable requirements as noted below. If the protected header field has been altered, the original value cannot be determined by the recipient. In addition, the encryption service cannot provide confidentiality for fields that must remain present in the message header during transport.

This document proposes a technology for securing message header fields. It's referred to as Secure Headers. It is based on S/MIME and CMS standards. It provides security services such as data integrity, confidentiality and non-repudiation of sender. Secure Headers is backward compatible with other

S/MIME clients. S/MIME clients who have not implemented Secure Headers technology need merely ignore specific signed attributes fields in a CMS signature (which is the default behavior).

4. Mechanisms to secure message header fields

Secure Headers technology involves the description of a security policy. This policy MUST describe a secure message profile and list the header fields to secure.

Secure headers are based on the signed attributes field as defined in CMS. The details are as follows. The message header fields to be secured are integrated in a structure (secure header structure) which is encapsulated in the signed attributes structure of the SignerInfo object. See Appendix A for an example. For each header field present in the secure signature, a status can be set. Then, as described in chapter 5.4 of CMS, the message digest calculation process computes a message digest on the content together with the signed attributes. Details of the signature generation process are described in chapter 4.5.1 of this document.

Verification of secure header fields is based on signature verification process described in CMS. At the end of this process, a comparison between the secure header fields and the corresponding message header fields is performed. If they match, the signature is valid. Otherwise, the signature is invalid. Details of the signature verification process are described in chapter 4.5.2 of this document.

Non-conforming S/MIME clients will ignore the signed attribute containing the secure headers structure, and only perform the verification process described in CMS. This guarantees backward compatibility.

Secure headers provide security services such as data integrity, non-repudiation and confidentiality.

For different reasons (e.g., usability, limits of IMAP [RFC 3501]), encryption and decryption processes are performed by a third party. The third party that performs these processes is referred to in Domain Security specification as a "Domain Confidentiality Authority" (DCA). Details of the encryption and decryption processes are described in chapters 4.6.1 and 4.6.2 of this document.

The architecture of Secure Headers is presented below. The MUA performs the signature generation process (C) and signature verification process (F). The DCA performs the message encryption process (D) and message decryption process (E). The encryption and decryption processes are optional.

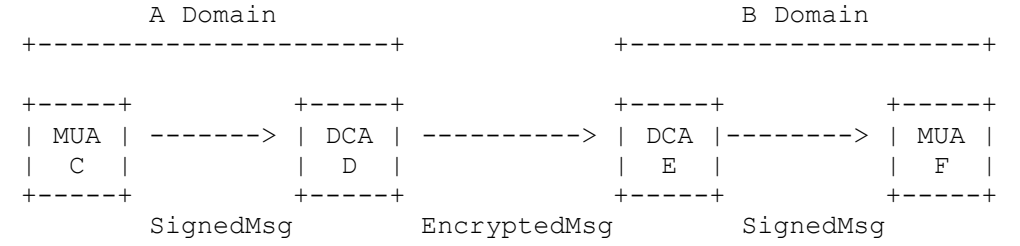


Figure 1: Architecture of Secure Headers

4.1. ASN.1 syntax of secure header fields

ASN.1 notation [X.680] of secure header structure is the follow:

```
 SecureHeaderFields ::= SET {
    canonAlgorithm Algorithm,
    secHeaderFields HeaderFields }
```

```
   id-aa-secureHeaderFieldsIdentifier OBJECT IDENTIFIER ::=
      {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
      pkcs-9(9) smime(16) id-aa(2) secure-headers (to be
      defined) }

   Algorithm ::= ENUMERATED {
      canonAlgorithmSimple(0),
      canonAlgorithmRelaxed(1) }

   HeaderFields ::= SET SIZE (1..max-header-fields) OF
      HeaderField

   max-header-fields INTEGER ::= MAX

   HeaderField ::= SEQUENCE {
      field-Name HeaderFieldName,
      field-Value HeaderFieldValue,
      field-Status HeaderFieldStatus DEFAULT duplicated }

   HeaderFieldName ::= VisibleString (FROM (ALL EXCEPT (":")))
        -- This description matches with the description of
        -- field name in the chapters 2.2 and 3.6.8 of RFC 5322

   HeaderFieldValue ::= IA5String (FROM (ALL EXCEPT (
      { 0, 0 }..{ 0, 9 } | { 0, 11 } | { 0, 12 } |
      { 0, 14 }..{ 1, 15 } | { 7, 15 })))
        -- This description matches with the description of
        -- field body in the chapter 2.2 of RFC 5322

   HeaderFieldStatus ::= INTEGER {
      duplicated(0), deleted(1), modified(2) }
```

4.2. Secure header fields length and format

This specification requires MUA security capabilities in order to process well formed headers, as specified in IMF. Notice that it includes long header fields and folded header fields.

4.3. Canonization algorithm

During a message transfer through a messaging system, some components might modify headers (i.e., space adding or deletion, lowercase/uppercase rewriting...). This might lead to header fields comparison mismatch. This emphasizes the need of a conversion process in order to transform data to their canonical form. This process is named canonization process.

Two canonization algorithms are considered here, according to DKIM specification [RFC 6376], chapter 3.4. The simple algorithm doesn't allow any modification whereas the relaxed algorithm accepts slight modifications like spaces replacement or line reformatting. Given the scope of this document, canonization mechanisms only involve header fields.

4.4. Header fields statuses

Header fields statuses are necessary to provide a confidentiality service toward message headers. In this specification, the confidentiality of header fields is provided by the DCA. This point is described in chapter 4. The DCA performs the message encryption process and message decryption process and these processes are described in details in the chapters 4.6.1 and 4.6.2. Although header fields statuses are embedded in the signature, the signature processes (generation and verification) ignore them. Since the confidentiality service is OPTIONAL, the status field is also OPTIONAL.

4.5. Signature Process

4.5.1. Signature Generation Process

During the signature generation process, the sender's MUA MUST embed the SecureHeaderFields structure in the signed

attributes, as described in CMS. SecureHeaderFields structure MUST include a canonization algorithm.

The sender's MUA MUST have a list of header fields to secure, statuses and a canonization algorithm, as defined by the security policy.

Header fields (names and values) embedded in signed attributes MUST be the same as the ones included in the initial message.

If different headers share the same name, all instances MUST be included in the SecureHeaderFields structure.

If multiple signatures are used, as explained in CMS and MULTISIGN [RFC 4853] specifications, SecureHeaderFields structure MUST be the same in each SignerInfos object.

If a header field is present and its value is empty, HeaderFieldValue MUST have a zero-length field-value.

Considering secure headers mechanisms, the signature generation process MUST perform the following steps:

1) Select the relevant header fields to secure. This subset of headers is defined according the security policy.

2) Apply the canonization algorithm for each selected header field. header field.

3) Complete the following fields in SecureHeaderFields structure according to the initial message: HeaderFieldName, HeaderFieldValue, HeaderFieldStatus (OPTIONAL).

4) Complete the algorithm field according to the canonization algorithm configured.

5) Embed the SecureHeaderFields structure in the signed attributes of the SignerInfos object.

6) Compute the signature generation process as described in CMS, chapter 5.5

4.5.2. Signature verification process

During the signature verification process, the receiver's MUA compares header fields embedded in the SecureHeaderFields structure with those present in the message. For this purpose, it uses the canonization algorithm identified in the signed attributes. If a mismatch appears during the comparison process, the receiver's MUA MUST invalidate the signature. The MUA MUST display information on the validity of each header field. It MUST also display the values embedded in the signature.

The receiver's MUA MUST know the list of mandatory header fields in order to verify their presence in the message. If a header field defined in a message is in the secure header list, it MUST be included in the SecureHeaderFields structure. Otherwise, the receiver's MUA MUST warn the user that a non-secure header is present.

Considering secure headers mechanisms, the signature verification process MUST perform the following steps:

1) Execute the signature verification process as described in CMS, chapter 5.6. If the signature appears to be invalid, the process ends. Otherwise, the process continues.

2) Read the type of canonization algorithm specified in SecureHeaderFields structure.

3) For each field present in the signature, find the matching header in the message. If there is no matching

header, the verification process MUST warn the user, specifying the missing header name. The signature is tagged as invalid.

4) Compute the canonization algorithm for each header field value in the message. If the simple algorithm is used, the steps described in DKIM, chapter 3.4.1, are performed. If the relaxed algorithm is used, the steps described in DKIM, chapter 3.4.2, are performed.

5) For each field, compare the value stored in the SecureHeaderFields structure with the value returned by the canonization algorithm. If values don't match, the verification process MUST warn the user. This warning MUST mention mismatching fields. The signature is tagged as invalid. If all the comparisons succeed, the verification process MUST also notify the user (i.e., using an appropriate icon).

6) Verify that no secure header has been added to the message header, given the initial fields. If an extra header field has been added, the verification process MUST warn the user. This warning MUST mention extra fields. The signature is tagged as invalid.

7) Verify that every mandatory headers in the security policy and present in the message are also embedded in the SecureHeaderFields structure. If such headers are missing, the verification process MUST warn the user and indicate the names of the missing headers.

The MUA MUST display features for each secure header field (name, value and status) and canonization algorithm used.

4.6. Encryption and Decryption Processes

Encryption and decryption operations are not performed by MUAs. This is mainly justified by IMAP limitations. The solution developed here relies on concepts explained in Domain Security specification, chapter 4. A fundamental component of the architecture is the Domain Confidentiality Authority (DCA). Its purpose is to encrypt and decrypt messages instead of (respectively) senders and receivers.

4.6.1. Encryption Process

All the computations presented in this chapter MUST be performed only if the following conditions are verified:

- The content to be encrypted MUST consist of a signature object or a multipart object, where one part is a detached signature, as shown in S/MIME specification, chapter 3.4.

- A SecureHeaderFields structure MUST be included in the signedAttrs field of the SignerInfo object of the signature.

All the mechanisms described below MUST start at the beginning of the encryption process, as explained in CMS. They are performed by the sender's DCA. The following steps MUST be performed for each field included in the SecureHeaderFields structure:

1. Extraction of the field status;

   1.1 If the status is Duplicated, the field is left at its existing value.

   1.2 If the status is Deleted, the header field (name and value) is removed from the message. Mandatory header fields specified in [RFC 5322] MUST be kept.

1.3 If the status is Modified, the header value is replaced by a new value, as configured in the DCA.

4.6.2. Decryption Process

All the computations presented in this chapter MUST be performed only if the following conditions are verified:

- The decrypted content MUST consist of a signature object or a multipart object, where one part is a detached signature, as shown in S/MIME specification, chapter 3.4.

- A SecureHeaderFields structure MUST be included in the SignerInfo object of the signature.

All the mechanisms described below MUST start at the end of the decryption process, as explained in CMS. They are executed by the receiver's DCA. The following steps MUST be performed for each field included in the SecureHeaderFields structure:

1. If the status is Duplicated, the field is left at its existing value.

2. If the status is Deleted, the DCA MUST write a header field (name and value) in the message. This header MUST be compliant with the information embedded in the signature.

3. If the status is Modified, the DCA MUST rewrite a header field in the message. This header MUST be compliant with the SecureHeaderFields structure.

5. Case of triple wrapping

Secure Headers mechanisms MAY be used with triple wrapping, as described in ESS [RFC 2634]. In this case, a SecureHeaderFields structure MAY be present in the inner

signature, in the outer signature, or both. In the last case, the two structure SecureHeaderFields MAY differ. One MAY consider the encapsulation of a header field in the inner signature in order to satisfy confidentiality needs. On the contrary, an outer signature encapsulation MAY help for delivery purpose. Header fields processing, given the signature type (inner or outer), is out of the scope of this document.

6. Security Gateways

Some security gateways sign or verify messages that pass through them. Compliant gateways MUST apply the process described in chapter 4.5.

For non-compliant gateways, the presence of SecureHeaderFields structure do not change their behavior.

In some case, gateways MUST generate new signature or insert signerInfos into the signedData block. The format of signatures generated by gateways is outside the scope of this document.

7. Security Considerations

This specification describes an extension of the S/MIME standard. It provides message headers integrity, non-repudiation and confidentiality. The signature and encryption processes are complementary. However, according to the security policy, only the signature mechanism MAY be prescribed. In this case, the signature process is implemented between MUAs. The encryption process requires signed messages with Secure Headers extension. If required, the encryption process is implemented by DCAs.

This specification doesn't address end-to-end confidentiality for message header fields. Sent and received messages by MUAs

MAY appear in plaintext. In order to avoid interception, the use of TLS is recommended between MUAs and DCAs (uplink and downlink). Another solution might be the use of S/MIME between MUAs and DCAs in the same domain.

For the header field confidentiality mechanism to be effective all DCAs supporting confidentiality must support SH processing. Otherwise, there is a risk in the case where headers are not obscured upon encryption, or not restored upon decryption process. In the former case confidentiality of the header fields is compromised. In the latter case the integrity of the headers will appear to be compromised.

8. IANA Considerations

IANA must register a suitable Object Identifier (OID) value for the identifier id-aa-secureHeaderFieldsIdentifier. This value will be used to identify an authenticated attribute carried within a CMS [RFC 5652] wrapper. This attribute OID appears in Section 4.1, and again in the reference definition in Appendix A. An appropriate registry arc is suggested in those instances of the draft text.

9. References

9.1. Normative References

[RFC 2045] Borenstein, N., "Multipurpose Internet Mail Extensions Part One", RFC 2045, November 1996.

[RFC 2119] Bradner, S., "Key words for use in RFCs to indicate requirement levels", RFC 2119, March 1997.

[RFC 2634] Hoffman, P., "Enhanced Security Services for S/MIME", RFC 2634, June 1999.

[RFC 4853] Housley, R., "Cryptographic Message Syntax (CMS), Multiple Signer Clarification", RFC 4853, April 2007.

[RFC 5322] Resnick, P., "Internet Message Format", RFC 5322, October 2008.

[RFC 5652] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 5652, September 2009.

[RFC 6376] Crocker, D., Hansen, T., Kucherawy, M., DomainKeys Identified Mail (DKIM) Signatures", RFC 6376, September 2011.

[X.680] ITU-T. Recommendation X.680 : Abstract Syntax Notation One (ASN.1): Specification of basic notation, November 2008.

9.2. Informative References

[RFC 2595] Newman, C., "Using TLS with IMAP, POP3 and ACAP", RFC 2595, June 1999.

[RFC 3183] Dean, T., Ottaway, W., "Domain security services using S/MIME", RFC 3183, October 2001.

[RFC 3207] Hoffman, P., "SMTP Service Extension for secure SMTP over Transport Layer Security", RFC 3207, February 2002.

[RFC 3501] Crispin, M., "Internet Message Access Protocol, version 4rev1", RFC 3501, March 2003.

[RFC 5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, October 2008.

[RFC 5598] Crocker, D., "Internet Mail Architecture", RFC 5598, July 2009.

[RFC 5750] Ramsdell, B., Turner, S., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Certificate Handling", RFC 5750, January 2010.

[RFC 5751] Ramsdell, B., Turner, S., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2, Message Specification", RFC 5751, January 2010.

Appendix A. Formal syntax of Secure Header

ASN.1 notation [X.680] of secure header structure is the follow:

```
SecureHeaderFields ::= SET {
     canonAlgorithm Algorithm,
     secHeaderFields HeaderFields }

id-aa-secureHeaderFieldsIdentifier OBJECT IDENTIFIER ::=
   iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-
   9(9) smime(16) id-aa(2) secure-headers (to be defined) }

Algorithm ::= ENUMERATED {
     canonAlgorithmSimple(0),
     canonAlgorithmRelaxed(1) }

HeaderFields ::= SET SIZE (1..max-header-fields) OF
   HeaderField

max-header-fields INTEGER ::= MAX

HeaderField ::= SEQUENCE {
     field-Name HeaderFieldName,
     field-Value HeaderFieldValue,
     field-Status HeaderFieldStatus DEFAULT duplicated }

HeaderFieldName ::= VisibleString (FROM (ALL EXCEPT (":")))
     -- This description matches with the description of
     -- field name in the chapters 2.2 and 3.6.8 of RFC 5322

HeaderFieldValue ::= IA5String (FROM (ALL EXCEPT (
   { 0, 0 }..{ 0, 9 } | { 0, 11 } | { 0, 12 } |
   { 0, 14 }..{ 1, 15 } | { 7, 15 })))
     -- This description matches with the description of
     -- field body in the chapter 2.2 of RFC 5322

HeaderFieldStatus ::= INTEGER {
     duplicated(0), deleted(1), modified(2) }
```

Appendix B. Secure Header Fields example

In the following example, header fields subject, from, to and x-ximf-primary-precedence are secured and integrated in a SecureHeaders structure.

Extract of message header fields

```
From: John Doe <jdoe@example.com>
To: Mary Smith <mary@example.com>
Subject: This is a test
X-ximf-primary-precedence: priority
```

SecureHeaders structure extracted from signature:

```
  2286  163:         SEQUENCE {
  2289   11:           OBJECT IDENTIFIER
                            '1 2 840 113549 1 9 16 2 80'
  2302  147:           SET {
  2305  144:             SET {
  2308    4:               ENUMERATED 1
  2314  135:                 SET {
  2317   40:                   SEQUENCE {
  2319   25:                     IA5String 'x-ximf-primary-
                                             precedence'
  2346    8:                     IA5String 'priority'
  2356    1:                     INTEGER 0
           :                     }
  2359   25:                   SEQUENCE {
  2361    2:                     IA5String 'to'
  2365   16:                     IA5String 'mary@example.com'
  2383    1:                     INTEGER 0
           :                     }
  2386   34:                   SEQUENCE {
  2388    4:                     IA5String 'from'
  2394   23:                     IA5String 'jdoe
                                            <jdoe@example.com>'
  2419    1:                     INTEGER 0
```

```
          :                       }
  2422   28:                     SEQUENCE {
  2424    7:                       IA5String 'subject'
  2433   14:                       IA5String 'This is a test'
  2449    1:                       INTEGER 0
          :                       }
          :                     }
          :                   }
          :                 }
          :               }
```

Example is displayed as an output of Peter Gutmann's "dumpasn1" program.

OID used in this example is non-official.

Appendix C. Acknowledgements

The authors would like to thank Alexey Melnikov, Damien Roque, Thibault Cassan, William Ottaway, and Sean Turner who kindly provided reviews of the document and/or suggestions for improvement. As always, all errors and omissions are the responsibility of the authors.

Authors' Addresses

Laurent CAILLEUX
DGA MI
BP 7
35998 RENNES CEDEX 9
France
Email: laurent.cailleux@intradef.gouv.fr

Chris Bonatti
IECA, Inc.
3057 Nutley Street, Suite 106
Fairfax, VA 22031
USA
Email: bonatti252@ieca.com

ANNEXE III - BREVET -

ARCHITECTURE ET PROCEDE DE TRANSMISSION D'UN COURRIEL

L'invention concerne le domaine des télécommunications et porte sur une architecture et un procédé de transmission d'un courriel entre un émetteur et un ou plusieurs destinataires.

Actuellement, les serveurs de messagerie de courrier électronique traitent de manière uniforme tous les courriels transmis entre un émetteur et un ou plusieurs destinataires. Ce principe de fonctionnement ne permet pas de différencier entre échanges de courriels personnels, professionnels, officiels, administratifs….

Il apparaît cependant qu'il existe un besoin de traitement différencié des échanges de courriels. Dans un environnement professionnel par exemple, il est souhaitable que le secret des correspondances soit garanti pour les courriels personnels, tandis que certains courriels professionnels doivent être signés de manière cryptographique ou autrement contrôlés en fonction d'une habilitation de l'émetteur et/ou du (ou des) destinataire(s).

Le brevet américain US 6 430 272 B1 divulgue un dispositif de commutation de messages, notamment des courriels, où les messages sont routés sur la base d'un ensemble de procédures de traitement des messages et en fonction de paramètres. Chaque utilisateur fixe les traitements qui seront réalisés en fonction de ses intentions. L'invention décrite dans le brevet américain US 6 430 272 B1 porte sur des mécanismes dédiés à la réception de messages et non à leur envoi (ou soumission). Les messages sont donc traités lors du processus de réception. Il n'est ainsi pas

possible, avec l'invention divulguée dans le brevet américain US 6 430 272 B1, de définir un traitement en amont, c'est-à-dire au niveau de l'envoi de messages, ce qui a pour conséquence que des messages, notamment des courriels, peuvent être envoyés et donc reçus au niveau d'un ou plusieurs destinataires sans avoir subi de traitement préalable, ce qui peut être préjudiciable par exemple en termes de sécurité.

La demande de brevet européen EP 1 003 308 A1 décrit un système de codage de priorité et de sécurité pour des courriels, qui positionne un code de caractérisation de courriel sur le courriel pour permettre aux serveurs de messagerie de destination de traiter plus efficacement le courriel. Le système de codage de priorité et de sécurité ne décrit pas la possibilité pour l'émetteur du courriel de lui affecter un code permettant d'effectuer un traitement spécifique sur le courriel avant son envoi à un routeur ou au serveur de messagerie de destination.

La demande internationale de brevet PCT/EP2007/057679 décrit un système et un procédé dans lesquels un utilisateur affecte à un courriel une indication de son contenu et définit des instructions de distribution du courriel à des destinataires en fonction de chacun des destinataires du courriel.

Le système et le procédé décrits dans la demande internationale de brevet PCT/EP2007/057679 ne permettent pas un traitement systématique des courriels en fonction uniquement du type qui lui a été attribué par l'émetteur. Dans le système et le procédé décrits dans la demande internationale de brevet PCT/EP2007/057679, l'émetteur doit entrer des informations sur chacun des destinataires, ce qui est fastidieux.

L'invention vise à pallier ces problèmes et a pour objectif de fournir une architecture et un procédé de

transmission de courriel aptes à appliquer des traitements spécifiques et appropriés à chaque courriel en fonction d'un attribut de traitement qui lui est affecté par l'émetteur.

L'objectif de l'invention est de fournir un système de messagerie électronique apte à appliquer des traitements spécifiques et appropriés à chaque courriel envoyé en fonction de son attribut (qui peut par exemple être son type, par exemple personnel, professionnel, administratif, opérationnel…). Lors de la soumission d'un courriel, le serveur de messagerie de l'émetteur transmet le message à un composant dédié qui réalise les différents contrôles, notamment d'autorisation. Après passage des contrôles, le composant dédié récupère les informations permettant d'identifier le serveur apte à prendre en charge les traitements spécifiques et appropriés à l'attribut de courriel et à lui appliquer avant un transfert vers le ou les destinataires.

L'invention peut être notamment mise en œuvre au moyen d'ordinateurs dans un environnement éventuellement virtualisé.

La présente invention a donc pour objet une architecture de transmission d'un courriel à un ou plusieurs destinataires, caractérisée par le fait qu'elle comprend :

- au moins un composant agent de soumission de message ;
- au moins un composant client de messagerie d'émetteur, par lequel un émetteur compose un courriel et lui affecte un attribut de traitement parmi un ensemble prédéfini de N attributs de traitement, le ou chaque composant client de messagerie d'émetteur étant en communication bidirectionnelle avec un composant agent de soumission de message ;

- N composants agent de transfert de message, chacun en communication bidirectionnelle avec le ou les composants agent de soumission de message et recevant en entrée un courriel, lui appliquant un traitement avant de l'émettre en sortie vers un ou plusieurs destinataires, chacun des N composants agent de transfert appliquant un traitement différent aux courriels qu'il reçoit, permettant autant de traitements différents de courriel qu'il y a d'attributs de traitement dans l'ensemble prédéfini d'attributs de traitement ;
- un composant répertoire de correspondance dans lequel est stockée une relation d'association entre chacun des N attributs de traitement de l'ensemble prédéfini des N attributs de traitement et chacun des N composants agent de transfert de message,

un composant d'interface entre chaque composant agent de soumission de message et le composant répertoire de correspondance qui, sur la base d'un attribut de traitement de courriel correspondant à un courriel reçu par un composant agent de soumission de message depuis un composant client de messagerie, interroge le composant répertoire de correspondance pour indiquer en retour audit composant agent de soumission de message le composant agent de transfert de message correspondant à l'attribut de traitement afin que le composant agent de soumission de message transfère ledit courriel audit composant agent de transfert de message correspondant.

Selon un mode de réalisation, l'architecture peut comprendre en outre un composant d'autorisation en communication bidirectionnelle avec le composant d'interface, le composant d'autorisation autorisant ou non la transmission d'un courriel entre un composant client de messagerie et un composant agent

de soumission de message en fonction de l'émetteur du courriel et/ou du (ou des) destinataire(s) et/ou de l'attribut de traitement qui lui a été affecté par l'émetteur, le composant d'autorisation autorisant en outre, lorsque le composant d'autorisation autorise le transfert du courriel entre le composant client de messagerie et le composant agent de soumission de message, le composant répertoire de correspondance à transmettre au composant d'interface dynamique l'indication du composant agent de transfert de message correspondant à l'attribut de traitement affecté au courriel par l'émetteur afin que le composant agent de soumission de message puisse transmettre le courriel à ce composant agent de transfert de message.

Le composant d'interface peut comprendre deux modules, l'un en communication bidirectionnelle avec le composant agent de soumission de message, l'autre en communication bidirectionnelle avec le composant répertoire de correspondance et le cas échéant avec le composant d'autorisation, les deux modules étant en communication bidirectionnelle. Ainsi, avec l'architecture de l'invention, un seul composant d'autorisation et un seul composant répertoire de correspondance peuvent être utilisés pour plusieurs composants agent soumission de messages. La configuration est ainsi beaucoup plus souple en cas de modification de politique ou d'ajout d'un nouvel attribut.

Le composant d'autorisation peut être un composant point d'exécution de politique en communication bidirectionnelle avec un composant point de décision de politique, selon le standard RFC2753.

L'invention a également pour objet un procédé de transmission d'un courriel d'un émetteur à un ou plusieurs destinataires, caractérisé par le fait qu'il comprend :

- la rédaction d’un courriel par l’émetteur ;
- l’affectation d’un attribut de traitement au courriel ;
- un traitement du courriel en fonction de l’attribut de traitement qui lui a été affecté par l'émetteur avant son transfert au destinataire ou aux différents destinataires.

Le procédé de transmission d'un courriel peut comprendre en outre une étape d'autorisation de transfert du courriel en fonction de l'émetteur du courriel et/ou du (ou des) destinataire(s) du courriel.

L’attribut de traitement peut comprendre un type de message et facultativement des informations supplémentaires.

Ces informations supplémentaires peuvent notamment être un niveau d’urgence ou un niveau de sensibilité du message.

Le procédé peut comprendre en outre une étape de vérification de l'autorisation du traitement du courriel correspondant à l’attribut qui lui a été affecté par l'émetteur avant son transfert au destinataire ou aux différents destinataires.

L'invention a également pour objet un serveur de messagerie, caractérisé par le fait qu'il comprend une architecture telle que définie ci-dessus ou qu'il met en œuvre un procédé tel que défini ci-dessus.

Le processus de soumission de courriel, selon un mode de réalisation particulier du procédé de l'invention, comporte les étapes suivantes :

- l'émetteur rédige un courriel sur son client de messagerie ;
- l'émetteur sélectionne un attribut pour le courriel (par exemple un type personnel, professionnel, administratif, opérationnel, spécifique entreprise…)
- l'émetteur envoie, à partir du client de messagerie, au composant de soumission de courriel du serveur de

messagerie, le courriel et l'attribut qui lui a été affecté par l'émetteur ;

- le courriel et son attribut sont reçus par le composant de soumission de courriel du serveur de messagerie ;
- en fonction de l'attribut, le composant de soumission de message transmet le courriel à un composant de transfert du serveur de messagerie qui sera en charge de lui appliquer une politique particulière ;
- le courriel auquel, avant son envoi, une politique particulière a été appliquée au niveau du serveur de messagerie est ensuite transmis à son ou ses destinataires.

L'attribut de message peut être constitué uniquement du type de message ou peut comporter le type et des informations supplémentaires permettant de caractériser plus finement l'échange de courriel entre l'émetteur et le ou les différents destinataires.

Comme indiqué précédemment, les informations supplémentaires peuvent comprendre notamment un niveau d'urgence ou un niveau de sensibilité du message.

Pour mieux illustrer l'objet de la présente invention, on va en décrire ci-après, à titre illustratif et non limitatif, un mode de réalisation particulier en référence au dessin annexé.

Sur ce dessin :

- la Figure 1 est un schéma d'un procédé de transfert d'un courriel selon un mode de réalisation de la présente invention.

Si l'on se réfère à la Figure 1, on peut voir que l'on y a représenté un mode de réalisation particulier de la présente invention.

Les termes et acronymes utilisés dans la description du système de l'invention et de ses flux sont extraits pour partie du standard RFC 5598 sur l'architecture de courriel sur internet (Internet Mail Architecture).

Le système comprend les éléments suivants :

- un composant client de messagerie (Mail User Agent - MUA) ;
- un composant agent de soumission de message (Mail Submission Agent - MSA) ;
- un composant agent d'exécution de correspondance (Correspondence Enforcement Agent – CEA) ;
- un composant agent de fourniture de correspondance (Correspondence Providing Agent – CPA) ;
- un composant point d'exécution de politique (Policy Enforcement Point – PDP) ;
- un composant point de décision de politique (Policy Decision Point - PDP) ;
- un composant répertoire de correspondance (Correspondence Repository – CR) ; et
- plusieurs composants agent de transfert de message (Message Transfer Agent – MTA).

Les composants MSA, CEA, CPA, PEP, PDP, CR et MTA sont des composants fonctionnels présents à l'intérieur du serveur de messagerie d'envoi de courriels. Ils peuvent également toutefois être distincts du serveur de messagerie et en communication avec celui-ci.

Du point de vue fonctionnel, le MUA est connecté uniquement au MSA par une interface de communication bidirectionnelle, le MSA est connecté d'une part au CEA et d'autre part à chaque MTA par une interface de communication bidirectionnelle, le CEA est connecté d'une part au MSA et d'autre part au CPA, également à chaque fois par une interface de communication bidirectionnelle, le CPA est connecté au PEP

et au CR à chaque fois par une interface de communication bidirectionnelle et enfin le PEP est connecté au PDP par une interface de communication bidirectionnelle. Les composants MSA, CEA, CPA, PEP, PDP, CR et MTA étant des composants dans le serveur de messagerie, les interfaces de communication bidirectionnelle entre ceux-ci sont des interfaces de communication logicielles.

Comme indiqué plus haut, ces composants (MSA, CEA, CPA, PEP, PDP, CR et MTA) peuvent également être en dehors du serveur de messagerie et en communication avec celui-ci, sans s'écarter du cadre de la présente invention.

Le procédé, selon le mode de réalisation décrit sur la Figure 1, est le suivant.

Dans une première étape, une connexion SMTP (Simple Mail Transfer Protocol) est ouverte entre le MUA et le MSA.

L'émetteur écrit son courriel dans le MUA et indique, toujours par l'intermédiaire du MUA, le type du courriel parmi plusieurs types de courriel prédéfinis.

Les types de courriel prédéfinis peuvent être par exemple : personnel, professionnel, administratif, opérationnel….

Le type de courriel affecté au courriel par l'émetteur est ensuite transmis du MUA au MSA.

Le MSA transmet une requête auprès du CEA, la requête comprenant :

- une demande d'autorisation d'envoi de courriel en fonction de l'émetteur voire du ou des destinataires et en fonction du type de courriel affecté ;
- une demande d'informations concernant le MTA correspondant au type de courriel affecté au courriel par l'émetteur, que le MSA devra contacter pour transmettre le courriel.

Le CEA transmet la requête au CPA, au format XML.

Le CPA s'interface avec une infrastructure de gestion des privilèges, puis transmet une requête d'autorisation au PEP.

Le PEP transmet une requête d'autorisation (au format XACML – eXtensible Access Control Markup Language) au PDP, qui transmet la réponse de requête d'autorisation, également au format XACML, au PEP.

Le PEP transmet la réponse de requête d'autorisation au CPA.

Si la réponse est négative, le CPA transmet l'information au MSA qui bloque le transfert du courriel à partir du MUA.

Si la réponse est positive (autorisation accordée par le PDP), le CPA transmet une requête de demande d'informations auprès du CR.

Le CR transmet alors au CPA les informations nécessaires au transfert différencié du courriel, c'est-à-dire notamment le MTA correspondant au type du courriel affecté par l'émetteur.

Le CPA transmet la réponse au CEA au format XML.

Le CEA transmet au MSA la réponse positive à la requête d'autorisation et les informations nécessaires au transfert différencié du courriel.

Le MSA, grâce aux informations récupérées, transfère le courriel au MTA correspondant qui appliquera la politique correspondant au type de courriel affecté par l'émetteur.

Le MSA transmet un code de prise en charge du courriel, et le MUA peut fermer la connexion SMTP ouverte dans la première étape s'il n'y a pas d'autres courriels à transmettre.

Comme indiqué plus haut, bien que le procédé ait été décrit en utilisant le seul type du message comme attribut, un attribut plus large comprenant le type du message et d’autres

facteurs permettant de caractériser plus finement l'échange peut être utilisé pour déterminer le MTA.

REVENDICATIONS

1 – Architecture de transmission d'un courriel à un ou plusieurs destinataires, caractérisée par le fait qu'elle comprend :

- au moins un composant agent de soumission de message (MSA) ;
- au moins un composant client de messagerie (MUA) d'émetteur, par lequel un émetteur compose un courriel et lui affecte un attribut de traitement parmi un ensemble prédéfini de N attributs de traitement, le ou chaque composant client de messagerie (MUA) d'émetteur étant en communication bidirectionnelle avec un composant agent de soumission de message (MSA) ;
- N composants agent de transfert de message (MTA), chacun en communication bidirectionnelle avec le ou les composants agent de soumission de message (MSA) et recevant en entrée un courriel, lui appliquant un traitement avant de l'émettre en sortie vers un ou plusieurs destinataires, chacun des N composants agent de transfert (MTA) appliquant un traitement différent aux courriels qu'il reçoit, permettant autant de traitements différents de courriel qu'il y a d'attributs de traitement dans l'ensemble prédéfini d'attributs de traitement ;
- un composant répertoire de correspondance (CR) dans lequel est stockée une relation d'association entre chacun des N attributs de traitement de l'ensemble prédéfini des N attributs de traitement et chacun des N composants agent de transfert de message (MTA),

un composant d'interface entre chaque composant agent de soumission de message (MSA) et le composant répertoire de correspondance (CR) qui, sur la base d'un attribut de traitement de courriel correspondant à un courriel reçu par un composant agent de soumission de message (MSA) depuis un

composant client de messagerie (MUA), interroge le composant répertoire de correspondance (CR) pour indiquer en retour audit composant agent de soumission de message (MSA) le composant agent de transfert de message (MTA) correspondant à l'attribut de traitement afin que le composant agent de soumission de message (MSA) transfère ledit courriel audit composant agent de transfert de message (MTA) correspondant.

2 – Architecture selon la revendication 1, caractérisée par le fait qu'elle comprend en outre un composant d'autorisation en communication bidirectionnelle avec le composant d'interface, le composant d'autorisation autorisant ou non la transmission d'un courriel entre un composant client de messagerie (MUA) et un composant agent de soumission de message (MSA) en fonction de l'émetteur du courriel et/ou du (ou des) destinataire(s) et/ou de l'attribut de traitement qui lui a été affecté par l'émetteur, le composant d'autorisation autorisant en outre, lorsque le composant d'autorisation autorise le transfert du courriel entre le composant client de messagerie (MUA) et le composant agent de soumission de message (MSA), le composant répertoire de correspondance (CR) à transmettre au composant d'interface dynamique l'indication du composant agent de transfert de message (MTA) correspondant à l'attribut de traitement affecté au courriel par l'émetteur afin que le composant agent de soumission de message (MSA) puisse transmettre le courriel à ce composant agent de transfert de message (MTA).

3 - Architecture selon la revendication 1 ou la revendication 2, caractérisée par le fait que le composant d'interface comprend deux modules, l'un en communication bidirectionnelle avec le composant agent de soumission de message (MSA), l'autre en communication bidirectionnelle avec le composant répertoire de correspondance (CR) et le cas

échéant avec le composant d'autorisation, les deux modules étant en communication bidirectionnelle.

4 - Architecture selon l'une des revendications 1 à 3, caractérisée par le fait que le composant d'autorisation est un composant point d'exécution de politique (PEP) en communication bidirectionnelle avec un composant point de décision de politique (PDP).

5 - Procédé de transmission d'un courriel d'un émetteur à un ou plusieurs destinataires, caractérisé par le fait qu'il comprend :

- la rédaction d'un courriel par l'émetteur ;
- l'affectation d'un attribut de traitement au courriel ;
- un traitement du courriel en fonction de l'attribut de traitement qui lui a été affecté par l'émetteur avant son transfert au destinataire ou aux différents destinataires.

6 – Procédé de transmission d'un courriel selon la revendication 5, caractérisé par le fait qu'il comprend en outre une étape d'autorisation de transfert du courriel en fonction de l'émetteur du courriel et/ou du (ou des) destinataire(s) du courriel.

7 – Procédé selon l'une des revendications 5 ou 6, caractérisé par le fait que l'attribut de traitement comprend un type de message et facultativement des informations supplémentaires.

8 – Procédé selon l'une des revendications 6 et 7, caractérisé par le fait qu'il comprend en outre une étape de vérification de l'autorisation du traitement du courriel correspondant à l'attribut qui lui a été affecté par l'émetteur avant son transfert au destinataire ou aux différents destinataires.

9 – Serveur de messagerie, caractérisé par le fait qu'il comprend une architecture selon l'une des revendications

1 à 4 ou qu'il met une œuvre un procédé selon l'une des revendications 5 à 8.

ABREGE

ARCHITECTURE ET PROCEDE DE TRANSMISSION D'UN COURRIEL

La présente invention a pour objet une architecture de transmission d’un courriel entre un émetteur et un ou plusieurs destinataires, caractérisée par le fait qu’elle comprend un composant client de messagerie (MUA) d’émetteur, un composant agent de soumission de message (MSA), N composants agent de transfert de message (MTA), un composant répertoire de correspondance (CR) dans lequel est stockée une relation d'association entre chacun de N attributs de traitement d'un ensemble prédéfini des N attributs de traitement et chacun des N composants agent de transfert de message (MTA), le composant client de messagerie (MUA) d’émetteur transmettant au composant agent de soumission de message (MSA), le courriel et un attribut de traitement qui lui a été affecté par l'émetteur, et le composant agent de soumission de message (MSA) transférant le courriel au composant agent de transfert de message (MTA) correspondant à l’attribut de traitement affecté au courriel par l'émetteur selon la relation d'association stockée dans le composant répertoire de correspondance.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% CREATION AUTOMATON
%
% .\automata\tools\TIOA\gui\workspace\COR
%
% V A.0.3
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% senderUserIdentity et recipientrMUAUserIdentity sont de la forme
addressrMUA
%

types

%corAttribute : Tuple[corAttributeName: String, corAttributeValue:
String],
%correspondence : Seq[corAttribute],

 addressrMUA : Tuple[localPart: String, domainPart: String],

 recipientrMUA: addressrMUA,

 MsgrMUA : Tuple[received: Seq[String], sender: addressrMUA,
recipientrMUAs: Seq[recipientrMUA], type : Enumeration[Message,
Bounce], date, rMUAMsgId: Nat, subject: String, content: String],

% Version integrant COR

% correspondence : Tuple[corName: String, corVersion: String],

% corMsgrMUA:  Tuple[received: Seq[String], sender: addressrMUA,
recipientrMUAs: Seq[recipientrMUA], type : Enumeration[Message,
Bounce], date,
%     rMUAMsgId: Nat, subject: String, content: String],
%     cor: correspondence,

 FlaggedMsgrMUArMUA : Tuple [message: MsgrMUA, flag: Bool]


automaton creation % rMUA

 signature

  % interfaces with user (submission)

   input writeMessage (message: MsgrMUA, senderUserIdentity:
addressrMUA, recipientUserIdentity: Seq[addressrMUA])
```

```
   output submitMessage (message: MsgrMUA, senderUserIdentity:
addressrMUA, recipientUserIdentity: Seq[addressrMUA])
   % Version integrant COR
   % output submitMessage (message: MsgrMUA, senderUserIdentity:
addressrMUA, recipientUserIdentity: Seq[addressrMUA], cor:
correspondence)

   output sendWarning % user receives a warning from this process

   internal localPolicyEnforcementrMUA

  states

   queueIn: Seq[MsgrMUA] := {};
   queueOut: Seq[MsgrMUA] := {};
   queueWarning: Seq[Nat] := {};
   localPolicyStatus : Bool :=false;

  transitions

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Recuperation du message cree par l'utilisateur
%% Attention -> assurer la verification de l'unicite des messages
%% Dans certains cas, date et MsgID sont positionnes
automatiquement par internal

   input writeMessage (message, senderUserIdentity,
recipientUserIdentity)
      eff queueIn := queueIn |- message;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
   internal localPolicyEnforcementrMUA
       pre queueIn ~= {};
       eff localPolicyStatus := choose status where status= true \/
status = false;
         if localPolicyStatus = true
          then queueOut := queueOut |- head(queueIn);
          else queueWarning := queueWarning |-
head(queueIn).rMUAMsgId;
         fi
     %
         queueIn := tail(queueIn);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   output  submitMessage (message, senderUserIdentity,
recipientUserIdentity)
      pre  queueOut ~= {};
      effmessage := head(queueOut);
    queueOut := tail(queueOut);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
output sendWarning
    pre queueWarning ~= {};
    eff % envoi d'un avertissement
        queueWarning := tail(queueWarning);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% SUBMISSION AUTOMATON
% V A.0.3
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% senderUserIdentity et recipientMSAUserIdentity sont de la forme
addressMSA
%

types
 %corAttribute : Tuple[corAttributeName: String, corAttributeValue:
String],
 %correspondence : Seq[corAttribute],

 addressMSA : Tuple[localPart: String, domainPart: String],

 recipientMSA: addressMSA,

 MsgMSA: Tuple[received: Seq[String], sender: addressMSA,
recipientMSAs: Seq[recipientMSA], type : Enumeration[Message,
Bounce], date, MSAMsgId: Nat, subject: String, content: String],

 FlaggedMsgMSAMSA : Tuple [message: MsgMSA, flag: Bool]

automaton submission % MSA

 signature

   input submitMessage (message: MsgMSA, senderUserIdentity:
addressMSA, recipientMSAUserIdentity: Seq[addressMSA])

   output initialTransferMessage (message:MsgMSA,
senderUserIdentity: addressMSA, recipientMSAUserIdentity:
Seq[addressMSA])
   % initialTransfer et finalTransfer

   internal localPolicyEnforcementMSA

  states

   queueInMSA: Seq[MsgMSA] := {};
   queueOutMSA: Seq[FlaggedMsgMSAMSA] := {};
   fMsgMSAMSA : FlaggedMsgMSAMSA;
   mesgMSA : MsgMSA;
   localPolicyStatusMSA : Bool ;
   postmasteraddressMSAMSA: addressMSA;

  transitions

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
   input submitMessage (message, senderUserIdentity,
recipientMSAUserIdentity)
% Attention -> ajouter la verification de l'unicite des messages en
arrivee

     effqueueInMSA := queueInMSA |- message;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   internal localPolicyEnforcementMSA
       pre queueInMSA ~= {} ;
       eff localPolicyStatusMSA := choose status where status= true
\/ status = false;
         % stockage du message et du statut associe
         fMsgMSAMSA.message := head(queueInMSA);
         fMsgMSAMSA.flag := localPolicyStatusMSA;
         queueOutMSA := queueOutMSA |- fMsgMSAMSA;
    %
         queueInMSA := tail(queueInMSA);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   output  initialTransferMessage (message, senderUserIdentity,
recipientMSAUserIdentity)
     pre  queueOutMSA ~= {} /\ fMsgMSAMSA = head(queueOutMSA);
     effmessage := fMsgMSAMSA.message;
      if fMsgMSAMSA.flag = true
    then % Envoi du message vers recipientMSAs
     message.type:=Message;
     %message.sender:=senderUserIdentity;
     %message.recipientMSAs:=recipientMSAUserIdentity;

     else % Envoi d'un bounce vers sender
       message.type:=Bounce;
       postmasteraddressMSAMSA.localPart := "postmasterMSA";
       postmasteraddressMSAMSA.domainPart := "domain";
       message.sender:= postmasteraddressMSAMSA;
       %
       message.recipientMSAs:= {};
       message.recipientMSAs:=message.recipientMSAs |-
senderUserIdentity;
   fi

   queueOutMSA := tail(queueOutMSA);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% TRANSFER AUTOMATON
% V A.0.3
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% senderUserIdentity et recipientMTAUserIdentity sont de la forme
addressMTA
%

types
 %corAttribute : Tuple[corAttributeName: String, corAttributeValue:
String],
 %correspondence : Seq[corAttribute],

 addressMTA : Tuple[localPart: String, domainPart: String],

 recipientMTA: addressMTA,

 MsgMTA: Tuple[received: Seq[String], sender: addressMTA,
recipientMTAs: Seq[recipientMTA], type : Enumeration[Message,
Bounce], date, MsgMTAId: Nat, subject: String, content: String],

 FlaggedMsgMTAMTA : Tuple [message: MsgMTA, flag: Bool]

automaton transfer % MTA

 signature

   input initialTransferMessage (message: MsgMTA,
senderUserIdentity: addressMTA, recipientMTAUserIdentity:
Seq[addressMTA])

   output finalTransferMessage (message:MsgMTA, senderUserIdentity:
addressMTA, recipientMTAUserIdentity: Seq[addressMTA])

   internal localPolicyEnforcementMTA

  states

   queueIn: Seq[MsgMTA] := {};
   queueOut: Seq[FlaggedMsgMTAMTA] := {};
   fMsgMTA : FlaggedMsgMTAMTA;
   localPolicyStatus : Bool ;
   postmasteraddressMTA: addressMTA;

  transitions

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   input initialTransferMessage (message, senderUserIdentity,
recipientMTAUserIdentity)
```

```
      effqueueIn := queueIn |- message;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   internal localPolicyEnforcementMTA
       pre queueIn ~= {} ;
       eff localPolicyStatus := choose status where status= true \/
status = false;
         % stockage du message et du statut associe
         fMsgMTA.message := head(queueIn);
         fMsgMTA.flag := localPolicyStatus;
         queueOut := queueOut |- fMsgMTA;
     %
         queueIn := tail(queueIn);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   output  finalTransferMessage (message, senderUserIdentity,
recipientMTAUserIdentity)
      pre  queueOut ~= {} /\ fMsgMTA = head(queueOut);
      effmessage := fMsgMTA.message;
       if fMsgMTA.flag = true
     then % Envoi du message vers recipientMTAs
      message.type:=Message;
      %message.sender:=senderUserIdentity;
      %message.recipientMTAs:=recipientMTAUserIdentity;

      else % Envoi d'un bounce vers sender
        message.type:=Bounce;
        postmasteraddressMTA.localPart := "postmasterMTA";
        postmasteraddressMTA.domainPart := "domain";
        message.sender:= postmasteraddressMTA;
        %
        message.recipientMTAs:= {};
        message.recipientMTAs:=message.recipientMTAs |-
senderUserIdentity;
    fi

    queueOut := tail(queueOut);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% DELIVERY AUTOMATON
% V A.0.3
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% senderUserIdentity et recipientMDAUserIdentity sont de la forme
addressMDA
%

types
 %corAttribute : Tuple[corAttributeName: String, corAttributeValue:
String],
 %correspondence : Seq[corAttribute],

 addressMDA : Tuple[localPart: String, domainPart: String],

 recipientMDA: addressMDA,

 MsgMDA: Tuple[received: Seq[String], sender: addressMDA,
recipientMDAs: Seq[recipientMDA], type : Enumeration[Message,
Bounce], date, MsgMDAId: Nat, subject: String, content: String],

 FlaggedMsgMDA : Tuple [message: MsgMDA, flag: Bool]

automaton delivery % MDA

 signature

   input finalTransferMessage (message: MsgMDA, senderUserIdentity:
addressMDA, recipientMDAUserIdentity: Seq[addressMDA])

   output deliveryMessage (message:MsgMDA, senderUserIdentity:
addressMDA, recipientMDAUserIdentity: Seq[addressMDA])

   internal localPolicyEnforcementMDA

  states

   queueIn: Seq[MsgMDA] := {};
   queueOut: Seq[FlaggedMsgMDA] := {};
   fMsgMDA : FlaggedMsgMDA;
   localPolicyStatus : Bool ;
   postmasteraddressMDA: addressMDA;

  transitions

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   input finalTransferMessage (message, senderUserIdentity,
recipientMDAUserIdentity)

      effqueueIn := queueIn |- message;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   internal localPolicyEnforcementMDA
       pre queueIn ~= {} ;
       eff localPolicyStatus := choose status where status= true \/
status = false;
         % stockage du message et du statut associe
         fMsgMDA.message := head(queueIn);
         fMsgMDA.flag := localPolicyStatus;
         queueOut := queueOut |- fMsgMDA;
     %
         queueIn := tail(queueIn);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   output  deliveryMessage (message, senderUserIdentity,
recipientMDAUserIdentity)
      pre  queueOut ~= {} /\ fMsgMDA = head(queueOut);
      effmessage := fMsgMDA.message;
       if fMsgMDA.flag = true
     then % Envoi du message vers recipientMDAs
       message.type:=Message;
       %message.sender:=senderUserIdentity;
       %message.recipientMDAs:=recipientMDAUserIdentity;

       else % Envoi d'un bounce vers sender
         message.type:=Bounce;
         postmasteraddressMDA.localPart := "postmasterMDA";
         postmasteraddressMDA.domainPart := "domain";
         message.sender:= postmasteraddressMDA;
         %
         message.recipientMDAs:= {};
         message.recipientMDAs:=message.recipientMDAs |-
senderUserIdentity;
    fi

    queueOut := tail(queueOut);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% RETRIEVING AUTOMATON
%
% V A.0.3
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% senderUserIdentity et recipientMSUserIdentity sont de la forme
addressMS
%

types
 % corAttribute : Tuple[corAttributeName: String,
corAttributeValue: String],
 % correspondence : Seq[corAttribute],

 addressMS : Tuple[localPart: String, domainPart: String],

 recipientMS: addressMS,

 MsgMS: Tuple[received: Seq[String], sender: addressMS,
recipientMSs: Seq[recipientMS], type : Enumeration[Message,
Bounce], date, MsgMSId: Nat, subject: String, content: String],

 userMsgMS : Tuple [userIdentity: addressMS, message: MsgMS,
userIdentity: addressMS, flag: Bool],

 userMsgMS2 : Tuple [message: Seq[MsgMS]],

 recordMailboxTable:Tuple [userIdentity: addressMS, index: Nat]

automaton retrieving %MS

 signature

  % Interfaces with MDA
   input deliveryMessage (message: MsgMS, senderUserIdentity:
addressMS, recipientMSUserIdentity: Seq[addressMS])

   % Delivery in mailbox
   output sendBounceMessage (message:MsgMS, senderUserIdentity:
addressMS, recipientMSUserIdentity: Seq[addressMS])

   internal localPolicyEnforcement

   % Interfaces with MUA
   input getMessage (userIdentity: addressMS)
   output retrieveMessage (messages: Seq[MsgMS])

  states

   queueInMS: Seq[MsgMS] := {};
```

```
   queueOutMS: Seq[MsgMS] := {};
   localPolicyStatus : Bool ;
   postmasteraddressMS: addressMS;

  % file des requetes de recuperation de messages
   queueUserRequest: Seq[addressMS] := {};
  % mailbox par utilisateur
  usersQueue: Seq[userMsgMS2] := {};
  % table des index par utilisateur
  mailboxTable: Seq[recordMailboxTable];

  %% Ajout dus aux limitations du simulateur Tempo
  u: Nat:=0;
  k: Nat:=0;

  transitions

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% input provenant du MDA pour remise des messages en mailbox (BAL)

   input deliveryMessage (message, senderUserIdentity,
recipientMSUserIdentity)
    %% les messages sont stockes en file avant l'application de la
politique dans localPolicyEnforcement
     eff queueInMS := queueInMS |- message;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% input provenant du MUA pour recuperer les messages

   input getMessage (userIdentity)
    %% stockage du nom du demandeur sous la forme addressMS
      eff queueUserRequest := queueUserRequest |- userIdentity;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% application de la politique interne
%% remise des messages dans les BAL des utilisateurs

   internal localPolicyEnforcement
       pre queueInMS ~= {} ;
       eff localPolicyStatus := choose status where status= true \/
status = false;
         %
         if localPolicyStatus = true
          then %% si l'application de la politique ne renvoie pas
d'erreur alors
           %% remise des messages par destinataire
           %
           %% len(head(queueInMS).recipientMSs : nombre de
destinataire par message
           %%

           %% modification du aux limitations du simulateur Tempo
```

```
        %%for  k: Nat where k > 0 /\ k <
(len(head(queueInMS).recipientMSs)) + 1
        k :=0;
        while k > 0 /\ k < (len(mailboxTable)) + 1

        do
         % recuperation de l'index de l'utilisateur dans la table
des mailbox
         % len(mailboxTable) : longueur de la table

         %% modification du aux limitations du simulateur Tempo
         %%for  u: Nat where u > 0 /\ u < (len(mailboxTable)) + 1
         while u > 0 /\ u < (len(mailboxTable)) + 1
          do
           if mailboxTable[u].userIdentity =
head(queueInMS).recipientMSs[k]
            %% Remise en BAL
            then usersQueue[u].message := usersQueue[u].message |-
head(queueInMS);
              % EXIT;
          fi
           od
      od

          else queueOutMS := queueOutMS |- head(queueInMS);
         fi
     %%
         queueInMS := tail(queueInMS);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% ouput : envoi des messages vers le MUA
%%
%% Attention -> Dans le modele suivant il n'y a pas de sauvegarde
de messages sur la MS

   output  retrieveMessage (messages)
     pre  queueUserRequest ~= {} ;
       eff% queueInMS := queueInMS |- message;
        % recuperation de l'index de l'utilisateur dans la table
des mailbox
     % len(mailboxTable) : longueur de la table

     messages := {};

     %% modification du aux limitations du simulateur Tempo
     %%for  u: Nat where u > 0 /\ u < (len(mailboxTable)) + 1
     u :=0;
     while u > 0 /\ u < (len(mailboxTable)) + 1
      do
       if mailboxTable[u].userIdentity = head(queueUserRequest)
        %%
```

```
        then %% boucle pour recuperer tous les messages d'un
utilisateur
          while usersQueue[u].message ~= {}
           do
            messages := messages |- head(usersQueue[u].message);
           %% suppression des messages dans usersQueue
           usersQueue[u].message := tail(usersQueue[u].message);
          od
           % EXIT;
      fi
       od

         %

         queueUserRequest := tail(queueUserRequest);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% ouput d'un avis d'erreur

    output sendBounceMessage (message, senderUserIdentity,
recipientMSUserIdentity)
    %parametres ???
      pre  queueOutMS ~= {} ;
      effmessage := head(queueOutMS);
        % Envoi d'un bounce vers sender
      message.type:=Bounce;
      postmasteraddressMS.localPart := "postmasterMS";
      postmasteraddressMS.domainPart := "domain1";
      message.sender:= postmasteraddressMS;
      %
      senderUserIdentity := postmasteraddressMS;
      %
      recipientMSUserIdentity := recipientMSUserIdentity |-
head(queueOutMS).sender;
      %
      message.recipientMSs := recipientMSUserIdentity;

    queueOutMS := tail(queueOutMS);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% DISPLAY AUTOMATON
%
% V A.0.3
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% senderUserIdentity et recipientMUAUserIdentity sont de la forme
addressMUA
%

types
 %corAttribute : Tuple[corAttributeName: String, corAttributeValue:
String],
 %correspondence : Seq[corAttribute],

 addressMUA : Tuple[localPart: String, domainPart: String],

 recipientMUA: addressMUA,

 MsgMUA: Tuple[received: Seq[String], sender: addressMUA,
recipientMUAs: Seq[recipientMUA], type : Enumeration[Message,
Bounce], date, MUAMsgId: Nat, subject: String, content: String],

 FlaggedMsgMUAMUA : Tuple [message: MsgMUA, flag: Bool]


automaton display % sMUA

 signature
  % interfaces with user (submission)
  % Cette action est a reetudier
  % Cette entree provient de l'environnement


   output displayMessage (message: MsgrMUA) % user display the
message

   internal localPolicyEnforcementMUA

   % interfaces with MS (Retrieving)
   output  getMessage (userIdentity: addressMUA)

   input retrieveMessage (messages: Seq[MsgMUA])

  states

   queueIn: Seq[MsgMUA] := {};
   queueOut: Seq[MsgMUA] := {};
   queueWarning: Seq[Nat] := {};
   localPolicyStatus : Bool ;
```

```
  transitions


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% requete vers le MS (IMAP)

   output  getMessage (userIdentity)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Recuperation des messages en provenance du MS suite a une
requete getMessage (IMAP)

   input retrieveMessage (messages)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Affichage, interpretation du message par l'utilisateur
   output  displayMessage (message)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   internal localPolicyEnforcementMUA
        pre queueIn ~= {};
        eff localPolicyStatus := choose status where status= true \/
status = false;
          if localPolicyStatus = true
           then queueOut := queueOut |- head(queueIn);
           else queueWarning := queueWarning |-
head(queueIn).MUAMsgId;
          fi
     %
          queueIn := tail(queueIn);
```

Résumé

Le service de courrier électronique en raison de sa simplicité d'utilisation combinée à son efficacité, a constitué l'un des principaux vecteurs de popularisation d'Internet. Il est devenu un service incontournable dont la richesse s'exprime au travers des usages variés et multiples qu'il autorise (privé, professionnel, administratif, officiel, militaire...). Cependant, toutes les réalisations existantes se réduisent techniquement à la mise en œuvre de politiques globales, compilant de façon statique un ensemble limité de fonctionnalités. Ces approches ne permettent pas au système de s'adapter de façon différenciée aux usages. De plus, le caractère rigide et monolithique de ces politiques peut parfois conduire à l'exécution inutile de traitements coûteux ou à l'impossibilité de satisfaire simultanément des exigences contradictoires.

Nous abordons cette problématique de l'évolution de la messagerie électronique dans le cadre général de la communication interpersonnelle d'un locuteur vers un interlocuteur. Nous identifions l'intention de communication du locuteur, comme un paramètre clé de toute communication interpersonnelle, dans la mesure où il permet de discriminer finement les communications réussies, parmi toutes celles qui sont comprises. Un second paramètre orthogonal au premier, défini comme le contexte du locuteur, s'avère déterminant lorsqu'il s'agit d'aborder la réalisation concrète des communications interpersonnelles réussies. La déclinaison de ces deux paramètres dans le cadre de la messagerie électronique nous conduit à concevoir la notion de correspondance. Cette dernière constitue une généralisation du courrier électronique dont la mise en œuvre offre une condition suffisante de qualification des échanges réussis, via ce média. Une correspondance permet de prendre en compte pour chaque message, l'intention de communication et le contexte de son émetteur. Sa mise en œuvre impose l'application en certains points du réseau, de politiques spécifiques au domaine administratif de référence, qui prennent en argument l'intention de communication et le contexte courant de l'émetteur. Un second bénéfice apporté par ce concept concerne le niveau de personnalisation du service de messagerie qui atteint une granularité de finesse maximale, du fait qu'il peut s'appliquer de façon différenciée, à chaque occurrence de message. Ces travaux ont abouti à la description d'une architecture représentative accompagnée de la définition de trois extensions de standards existants (SUBMISSION, IMF et S/MIME). Notre approche a été illustrée à travers deux cas d'usages importants, conformes à des spécifications recommandées pour les domaines administratif (RGS- référentiel général de sécurité) et militaire (MMHS - Military Message Handling System).

Mots-clés : Messagerie électronique, Messagerie sécurisée, Messagerie de confiance, Sécurité, Politique, Correspondance électronique

Abstract

The ease of use and efficiency of the email service contributed to its widespread adoption. It became an essential service and authorizing multiples and various uses (private, professional, administrative, governmental, military ...). However, all existing systems are technically reduced to the implementation of global policies, compiling in a static way a limited set of features. These approaches prevent differentiated adaptations of the system to the uses. The rigid and monolithic nature of these policies can moreover lead to unnecessary execution of expensive treatments or to the inability to simultaneously satisfy conflicting requirements.

We address this problem of the evolution of e-mail in the general context of interpersonal communication of a sender to a receiver. We identify the sender's intention of communication, as a key parameter of any interpersonal communication, insofar as it allows to finely discriminate the successful communications, between all the ones that are understood. A second parameter which is orthogonal to the first, defined as the context of the sender, is important because it allows to determine the successful aspect of an interpersonal communication. The declination of these two parameters in the electronic mail led us to define the concept of electronic correspondence. This one is a generalization of the email the implementation of which provides a sufficient condition of qualification successful exchanges via this medium. A correspondence allows taking into account for each message, the intention of communication and context of its sender. Its implementation requires in certain points of the network, the enforcement of specific policies depending of an administrative domain and which take as argument the intention of communication and the current context of the sender. A second benefit provided by this concept concerns the level of customization of messaging reaching a maximum granularity, because it can be applied in a differentiated way, to each message instance. These works led to the description of a representative architecture and the definition of three extensions to existing standards (SUBMISSION, IMF and S/MIME). Our approach has been illustrated through two main use cases, compliant with recommended specifications for administration (RGS - Référentiel Général de Sécurité) and military (MMHS - Military Message Handling System) domains.

Keywords : Electronic mail, Secure email system, Trust email system, Security, Policy, Electronic correspondence

TELECOM Bretagne

n° d'ordre : 2015telb0335

Télécom Bretagne
Technopôle Brest-Iroise - CS 83818 - 29238 Brest Cedex 3
Tél : + 33(0) 29 00 11 11 - Fax : + 33(0) 29 00 10 00